COM ESTA EDIÇÃO TERMINA O PRIMEIRO VOLUME DE MICROCOMPUTADOR CURSO PRÁTICO
ADQUIRA A CAPA DURA PARA ENCADERNÁ-LO.

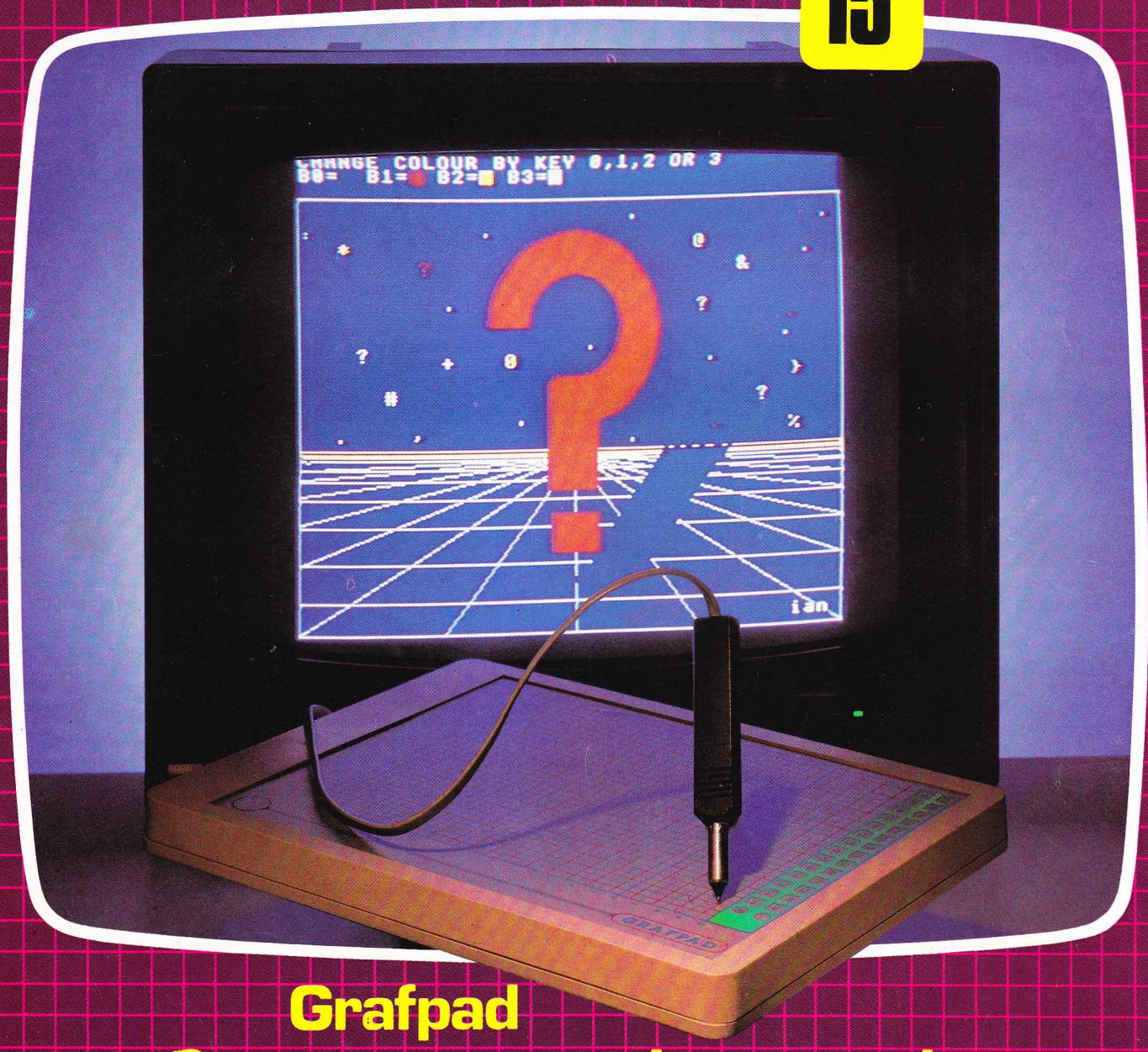
MICROCOMPUTADOR CURSO PRÁTICO

TIRE O MÁXIMO DO MICRO EM 30 SEMANAS

PUBLICAÇÃO SEMANAL ILUSTRADA

15

Cr$ 8.200



Grafpad

Como comprar micros usados

rioGráfica

MANAUS, ALTAMIRA, MACAPÁ, JI-PARANÁ, PORTO VELHO, RIO BRANCO, SANTARÉM, BOA VISTA, SINOP, ALTA FLORESTA: Cr$ 10.660

MICROCOMPUTADOR CURSO PRÁTICO

APLICAÇÕES

Túnel do tempo 297

PROJETOS DE PROGRAMAS

Teste de reflexos 300

PERIFÉRICOS

O melhor de dois mundos 301

SOFTWARE

Organização de dados 303

RAIO X

Grafpad 305

TÉCNICAS DE PROGRAMAÇÃO

Check-up de rotina 308

CHIPS & BYTES

Segunda mão 310

LINGUAGENS

Lingüística estrutural 312

Sumário 315

TESTE

Depois de ler com atenção todo o fascículo, responda a estas perguntas. Veja a solução no próximo número, junto com um novo teste.

1) Qual a diferença, em BASIC, entre as variáveis TOTAL e TOTAL%?
2) Na escolha de um micro, qual o fator a ser considerado prioritariamente: o hardware ou o software?
3) Quais os três tipos de sinais que produzem imagens no vídeo de um micro?
4) Como se chama a folha plástica que se coloca sobre um tablete digitalizador, desenhando nela comandos específicos para outras aplicações?

RESPOSTAS DO TESTE ANTERIOR

1) O dispositivo essencial para que um micro controle aparelhos eletromecânicos é o relé.
2) Um printer/plotter desenha os caracteres de texto como se fossem padrões gráficos.
3) Um número real, com sua parte fracionária, ocupa 4 ou 5 bytes, ao passo que um número inteiro utiliza apenas 2. Isto influi na velocidade de execução do programa.
4) O bit acrescentado a um código binário para assegurar a correção da transmissão dos dados chama-se "bit de paridade".

PLANO DA OBRA

MICROCOMPUTADOR - CURSO PRÁTICO é uma coleção de 30 fascículos de periodicidade semanal. Os 29 primeiros terão, cada um, vinte páginas internas (miolo) e quatro capas, além de uma introdução de dezesseis páginas acompanhando o fascículo 1. O fascículo 30 incluirá o índice geral da obra. Os miolos, encadernados, constituem dois volumes de um curso prático de microcomputação, com seções indicadas por tarjas de diferentes cores.
As duas primeiras capas são descartáveis; as 3.ª e 4.ª trazem programas de jogos. Encadernadas, formam um volume de cem páginas.

COMO ENCADERNAR

As capas duras, incluindo as guardas, estarão à venda simultaneamente com os fascículos 10 (**Volume 1**) e 30 (**Volume 2** e **Jogos**).
Volume 1 - Deve ser encadernado com os elementos dispostos nesta ordem: guardas; frontispício e introdução (publicados com o fascículo 1); miolos dos fascículos de 1 a 15; e guardas.
Volume 2 - Guardas; frontispício (que virá com o fascículo 30); miolos dos fascículos de 16 a 30; e guardas.
Jogos - Guardas; frontispício e sumário (que constituem as 3.ª e 4.ª capas do fascículo 30); 3.ª e 4.ª capas dos fascículos de 1 a 29; e guardas.

rioGráfica **Diretoria Executiva:** Oscar Neves; Nilo Sérgio de Almeida; Danilo Esteves Costa; **GRUPO EDUCAÇÃO E CULTURA - Planejamento e Comercialização - Gerente de Planejamento:** João Azeredo Coutinho; **Gerente de Marketing:** Jaime Rodrigues; **Gerente de Produto:** José Mauro Porto; **Gerente do P.C.P.:** Aylton Menezes; **Diretor de Vendas:** Marcos March; **Gerente de Vendas:** Rubens Barbosa; **Coordenadores de Promoção:** Paulo Cesar M. Seixas; Edgar Mello M. Neto; **Gerente de Circulação:** Norberto Martin; **Supervisor de Operações:** Abel Pereira Pinto; **Editorial - Diretor:** João J. Noro; **Editor-chefe:** Maurício Rittner; **Administração Editorial:** Heitor de Souza Paixão; **Editor:** Jesse Navarro; **Revisores:** Isis Augusta Loyolla; Cacilda Guerra; Luiz Vicente Vieira Filho; Marcos Domingos Agathão; Maria da Graça Mendonça Couto; **Diretor de Arte:** Adalberto Cornavaca; **Chefe de Arte:** Bonifácio Duardes Miranda; **Chefe de Estúdio:** José Yuji Kuribayashi; **Assistente de Arte:** Miguel Luis Escámez Simón; **Colaboradores - Editores-assistentes:** Claudio Marcondes; Isa Mara Lando; Luiz Carlos Pizarro Marin, **Texto:** Carlos Eduardo Silveira Matos; **Preparação de Texto:** Luiz Carlos Cardoso; **Tradução:** Fernando Dorfman Knijnik; Lúcia Maria Leal Gonçalves; Maria Clara Cescato; Maria Ester Menezes Martinho; Newton Roberval Eichemberg; Patrícia de Paiva e Castro; Regina Amarante; **Pesquisa:** Stella Maria Quentel; **Arte:** Edemilson Ribeiro da Silva; **Assessoria Técnica:** Geraldo Coen; Gustavo José Ferreira Grubba; Lauro de Lauro; Pierluigi Piazzi; Sérgio Jorge Batarce. As fotos não creditadas pertencem à obra original. Seguem-se os créditos das demais: Fernando Scavone. © APSIF Copenhagen, 1984; © Orbis Publishing Ltd., 1984; © Editora Rio Gráfica Ltda., 1985, para a língua portuguesa. **Composição:** Editora Rio Gráfica Ltda. e AM Produções Gráficas Ltda. **Impressão:** JBIG.

NÚMEROS ATRASADOS - Você poderá comprar os exemplares que faltam em sua coleção, pelo preço do último fascículo posto à venda, em sua banca de jornal preferida. Caso não consiga obtê-los, faça sua solicitação por carta endereçada diretamente à Editora Rio Gráfica Ltda., rua Itapiru, 1209, Rio Comprido, Rio de Janeiro, CEP 20251. O atendimento será feito por via postal. Nas cidades do Rio de Janeiro e de São Paulo, as compras poderão ser efetuadas pessoalmente nos seguintes endereços: Rio de Janeiro - Rua Itapiru, 1209, Rio Comprido; São Paulo - Rua Frei Caneca, 1152, Consolação.

Distribuidor exclusivo para todo o Brasil: Fernando Chinaglia - Rua Teodoro da Silva, 907, Rio de Janeiro. Distribuidor para Portugal: Electroliber Lda. - Rua Prof. Reinaldo dos Santos, 1488, Lisboa.

TÚNEL DO TEMPO

Na escola, um micro pode criar meios ambientes, sem limitações de tempo e espaço. Seu enorme potencial para a educação aumentou ainda mais com os jogos de simulação e aventura.

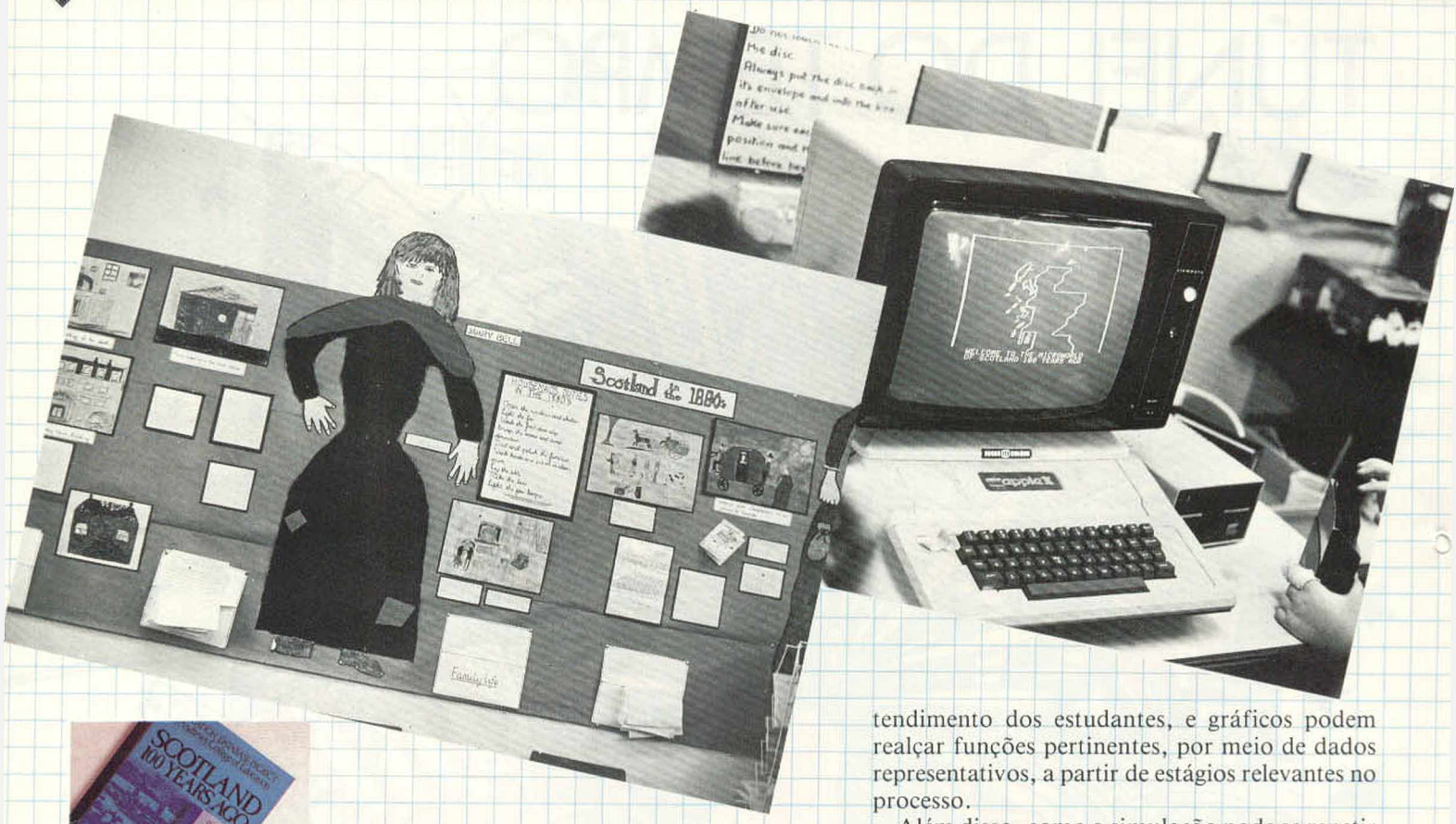
Existe alguma razão para o aprendizado não ser tão divertido quanto os jogos? Todas as crianças — e a maioria dos professores — responderão negativamente à pergunta. O uso dos jogos (em especial dos dramáticos, com desempenho de papéis) adquire cada vez mais popularidade, acompanhando o desenvolvimento das técnicas de dramatização na educação e da teoria dos jogos na psicologia. Tais desenvolvimentos deram ainda mais crédito ao potencial dos computadores nas escolas.

Os videojogos de aventura e simulação do tipo Invasores do Espaço e outros mais recentes representam as duas categorias predominantes desse tipo de diversão. E, à medida que o software se desenvolve, torna-se mais difícil distinguir as diferenças entre elas; no entanto, a classificação persiste.

Apesar de sua popularidade, os jogos do gênero guerra espacial tiveram pouco impacto sobre o software educacional. Ao escreverem o livro *Mind at play, the psychology of video games (O pensamento em jogo, a psicologia dos videojogos),* Elizabeth e Geoffrey Loftus perguntaram a vários psicólogos qual a eficácia educacional desses jogos. As respostas não foram encorajadoras. O valor de entretenimento do tipo "acabar-com-eles" ultrapassa em muito qualquer valor educativo. No entanto, a pesquisa mostrou que, devido à coordenação e aos reflexos exigidos, esse tipo de jogo pode ter efeitos terapêuticos sobre os portadores de vários tipos de desequilíbrio mental e de lesões no cérebro.

Houve tentativas de reforçar o conteúdo educacional desses jogos: ao soletrar corretamente uma palavra, por exemplo, a criança abatia os invasores. Todavia, comprovou-se que os jogos modificados eram menos divertidos que os originais — e o benefício educacional era, na melhor das hipóteses, duvidoso.

Aventuras na escola
Os jogos educacionais de aventura e simulação tornam possível à criança explorar, do interior da sala de aula, situações de outra forma inacessíveis ou mesmo potencialmente perigosas. A experiência tem mostrado que esses jogos, longe de ser uma perda de tempo, podem acrescentar muito ao processo de aprendizado.



Viagem no tempo
O Projeto Simulação-Banco de Dados usa o computador como ferramenta suplementar, destinada a atuar juntamente com outras atividades de aprendizado. O pacote de programas divide-se em três partes: um jogo de aventuras, um conjunto de arquivos de texto que fornece informações básicas e um banco de dados contendo outras referências e fontes. Neste exemplo, a turma de escolares foi dividida em seis grupos e cada um assumiu o papel de um personagem em viagem pela Escócia do final do século XIX. As atividades decorrentes do uso do banco de dados incluíram redação criativa, cartografia, dramatização e excursões pela Escócia contemporânea.

Em contrapartida, os jogos de simulação e de aventura apresentam grande potencial como alternativas educacionais. Os primeiros usam o computador para criar instrumentais de aprendizagem até então inacessíveis à criança; e os de aventura reproduzem cenários fantásticos, sons e efeitos especiais das fábulas e da ficção científica. Esses princípios já são aproveitados nos programas de treinamento industrial, e hoje se empregam muitos deles em escolas.

Ballooning (Uma Viagem de Balão) é um jogo de simulação típico, criado na Grã-Bretanha. Na parte inferior da tela há um painel de instrumentos, com altímetro, marcador de combustível, de temperatura, e um indicador da velocidade de descida ou de ascensão, que reage conforme se acionam o queimador e a válvula de gás. Quando o balão desce a certa altura, o terreno torna-se visível; caso haja necessidade de abastecimento, existem vários postos terrestres para os quais o balão deve ser conduzido.

Vantagens educacionais

Um computador pode apresentar em detalhe a estrutura física de qualquer experimento ou exercício. As técnicas de simulação permitem ao aluno concentrar-se apenas na situação em desenvolvimento, no que está acontecendo e no porquê. O tempo deixou de ser um obstáculo à finalização, em sala de aula, de projetos complexos e longos: experiências que exigiram dias no laboratório surgem simuladas na tela em segundos. Um modelo em computador simplifica sistemas que seriam complexos demais para o entendimento dos estudantes, e gráficos podem realçar funções pertinentes, por meio de dados representativos, a partir de estágios relevantes no processo.

Além disso, como a simulação pode se repetir sob várias condições, não há limite para o número de experimentos à disposição dos estudantes. Estes controlam as variáveis implementadas e os rumos que o experimento toma, o que lhes proporciona melhor compreensão dos resultados e das conclusões.

Mas existem limites ao uso da simulação. Como o computador só pode armazenar um número predefinido de variáveis, há o risco de uma simulação apresentar visão por demais simplificada do tema. Assim, embora seja seu maior atributo, a capacidade de representar situações reais é também sua maior falha.

Um programa nos moldes do Ballooning, com número fixo de variáveis, atinge seu maior benefício quando usado, num contexto mais amplo, como um entre muitos aspectos da ciência e da história da aeronáutica. Mas muitas vezes se ignora que a simulação não substitui a realidade. Sua força está em estimular o interesse por temas associados e não em representar uma extensão das técnicas de ensino programado.

Exemplo extremo de emprego questionável de computador é o programa que simula um circuito ligando lâmpadas e baterias de várias maneiras. Ao que parece, foi desenvolvido para uma escola capaz de adquirir um computador caro, mas que, por algum motivo misterioso, não podia comprar pilhas e lâmpadas.

Criando micromundos

Nos últimos anos, livros e filmes como *O senhor dos anéis, 2001: Uma odisséia no espaço* e *Conan, o bárbaro,* conquistaram enorme popula-

ridade. Nada estranho, visto que a maioria das pessoas — crianças e adultos — gosta de abandonar-se à fantasia. E não é por outro motivo que os jogos de aventura se tornaram enorme sucesso comercial, além de valorizarem o desempenho de papéis dramáticos — o que os especialistas chamam de "role playing" — na terapia e na educação.

A natureza interativa do computador faz dele uma ferramenta ideal para os jogos de fantasia e de aventura. Suas possibilidades gráficas e sonoras podem dar "realidade" a um mundo fantástico ou a uma época do passado. E os educadores logo se apossaram dessa capacidade de criar "micromundos", vendo-os como um salto qualitativo nas práticas de ensino.

Aparentemente, não haveria melhor maneira de uma criança aprender história do que viajando ao passado e vivenciando uma aventura. Mas esse "viajante no tempo" necessitaria de informações para sobreviver; assim, o programa deveria conter um banco de dados que servisse de fonte de consulta durante o jogo.

O Projeto Simulação-Banco de Dados (Simulation Database Project) foi elaborado por Allan Martin no St. Andrews College of Education, de Glasgow, "para explorar a forma pela qual os computadores podem contribuir de modo significativo para o trabalho na escola primária, criando um novo tipo de objeto pedagógico baseado no computador".

A estrutura da simulação lembra bastante um livro de aventuras: a criança ingressa num mundo histórico, viaja por diferentes locais e enfrenta várias situações. Os arquivos de texto no banco de dados estão relacionados a cada local e são acessíveis a qualquer momento. Um segundo banco de dados contém detalhadas referências ao local ou assunto e pode ser pesquisado para informações complementares em livros, filmes e outros meios de comunicação. A maior parte do aprendizado corre independentemente do computador, que é apenas um entre vários recursos educacionais.

Para melhor integração do programa a outras atividades em classe, o software inclui um manual, um mapa da área, cartas, ilustrações e outros materiais de consulta. Os programas podem ser ajustados a condições locais e são escritos em LOGO — linguagem utilizada por um número crescente de escolas brasileiras e que, segundo os especialistas, facilita o desenvolvimento do raciocínio lógico e desperta a criatividade infantil, constituindo uma porta de entrada ideal para o mundo dos computadores.

Um bom exemplo de banco de dados simulado tem por tema a Escócia da década de 1880. O programa foi usado com uma turma dividida em seis grupos e cada qual assumiu a identidade de um personagem fictício — mas autêntico — representativo de uma camada social: aristocratas, industriais, religiosos, operários, empregados domésticos etc. Cada grupo recebeu um roteiro e era obrigado a viajar pela Escócia. Após planejar a viagem com o mapa e chegar ao primeiro local, o grupo-personagem deparava-se com vários incidentes. Depois de cada fase de movimento, as crianças, usando dois bancos de dados e outras fontes de consulta fornecidas pelo professor, investigavam o local. As atividades derivadas dessas investigações incluíam cartografia, elaboração de diários e reportagens para jornais, desenhos e pinturas do ambiente. Além disso, as crianças produziram peças, gravaram "entrevistas" com personagens da época, visitaram as áreas abrangidas pelo jogo e convidaram oradores alheios à escola para realizarem conferências sobre a Escócia.

O Projeto Simulação-Banco de Dados exemplifica como um videojogo pode ser usado para interessar alunos e estimular o trabalho em equipe. Dois outros pacotes deste mesmo projeto começaram a ser elaborados em 1984: Palestina no Tempo de Cristo, empregado em cursos histórico-religiosos, e Os Rios, subsídio aos estudos do meio ambiente.

Essas simulações empregando as excepcionais qualidades do computador produzem um novo e fascinante recurso educacional. O micro torna-se um meio pelo qual a criança pode adquirir uma experiência valiosa, entrando em contato com ambientes e épocas históricas até então inacessíveis — outros mundos.

Homo ludens

Um programa na tevê britânica mostrou um grupo de crianças desenhando uma tartaruga mecânica guiada pela linguagem LOGO no chão da sala de aula. O locutor observou aos telespectadores que eles não deviam se iludir com o caráter improvisado do exercício, afirmando: "De fato, aqui está ocorrendo aprendizado real".

Tais palavras certamente perturbaram os pais e professores que acreditam na tradicional diferenciação entre "brincadeira" e "estudo sério". A verdade, porém, é que os limites entre essas categorias tendem a diluir-se — e muitos acreditam que o aprendizado "real" ocorre somente através dos jogos.

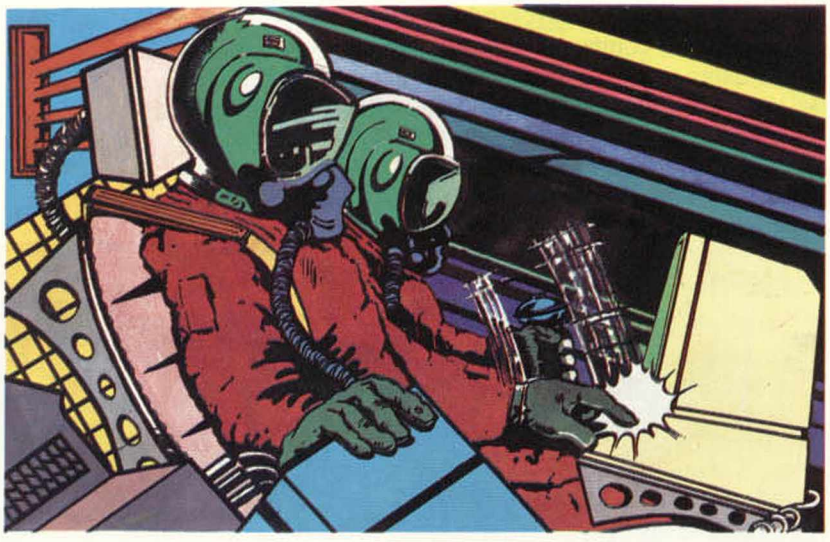
Há uma base para esse ponto de vista. Em seu *Homo ludens,* lançado em 1944, o pensador holandês J. Huizinga afirma que, além de processo de aprendizado, o jogo tem valor substancial na cultura humana por si mesmo. Eis alguns de seus argumentos:

- Jogar é uma função significativa, dotada de sentido.
- O jogo é voluntário e instintivo. As crianças brincam para aprender sobre o mundo "adulto", mas continuam a fazê-lo por ser divertido.
- A civilização contemporânea sofre de "formalização de jogos": quanto mais estruturados, menos são jogos.

TESTE DE REFLEXOS

Crianças e adolescentes conseguem melhor desempenho em jogos de ação, onde uma resposta rápida é tudo o que importa. O jogo aqui apresentado permite que você meça seu tempo de reação desviando-se de asteróides.

Todos gostamos de pensar que nossos reflexos são ótimos, que nossas reações aos eventos acontecem instantaneamente. No entanto, reagindo a um estímulo, a resposta mais rápida de que somos capazes demora cerca de um terço de segundo. Parece pouco, mas tal defasagem pode ser fatal. Um carro em alta velocidade, por exemplo, percorre quase 10 m nesse tempo.



Os tempos de reação variam muito de uma pessoa para outra, e álcool, cansaço ou doença exercem efeito desfavorável sobre os resultados. O programa Socorro Espacial permite testar os reflexos de qualquer pessoa e, para tornar os tempos mais precisos, considera a média obtida em cinco tentativas.

Neste jogo, você pilota uma espaçonave e precisa levar com urgência sua carga de medicamentos a certa colônia instalada num cinturão de asteróides. A velocidade é vital para que o socorro chegue a tempo, mas a nave deve ser desviada sempre que houver perigo de colisão. Acionando a barra de espaço, você pára a espaçonave, evitando bater nos asteróides. O jogo mostra o tempo transcorrido entre o momento em que o asteróide aparece na tela e o instante em que se pressionou a barra de espaço. Após cinco paradas da nave, a média do tempo de reação é calculada.

Para se certificar de que o jogo vem transcorrendo com seriedade, o programa verifica se a barra de espaço não está sendo pressionada continuamente. Se estiver, os motores serão desligados e soará um alarme. Quando os asteróides surgem no vídeo, o radar da nave emite um som. Se quiser dificultar o exercício, mude o programa de modo a receber um alerta auditivo ou visual, mas não os dois.

Socorro Espacial

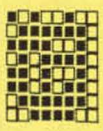
TK 85

```
  10 REM ▚▞ PARA O TK85 ▚▞
  20 LET M=0
  30 LET C=0
  40 LET V=0
  50 DIM R(5)
  60 POKE 16418,0
  70 CLS
  80 FOR G=1 TO 5
  90 FOR P=1 TO RND*30+10
 100 PRINT AT INT (RND*22),INT (
RND*32);"."
 110 IF INKEY$="" THEN GOTO 150
 120 PRINT AT 21,0;"AGUARDE O AS
TEROIDE"
 130 IF INKEY$<>"" THEN GOTO 130
 140 PRINT AT 21,0;"
                   "
 150 NEXT P
 160 PRINT AT 11,16;"■"
 170 POKE 16436,255
 180 FAST
 190 LET P=P+1
 200 IF INKEY$<>"P" THEN GOTO 18
0
 210 LET R(G)=10+10*(P-1110)/110
0
 220 PRINT AT 22,0;"VOCE DEMOROU
 ";INT (R(G)*100)/100;"SEGUNDOS
PARA PARAR OS MOTORES."
 230 SLOW
 240 LET M=M+R(G)
 250 FOR L=1 TO 150
 260 NEXT L
 270 CLS
 280 NEXT G
 290 LET A=M/5
 300 CLS
 310 PRINT "APOS 5 TENTATIVAS, S
UA MEDIA DE TEMPO DE REACAO FOI
DE ";INT (A*100)/100,"SEGUNDOS"
 320 FOR G=1 TO 5
 330 LET V=V+ABS (R(G)-A)
 340 NEXT G
 350 PRINT
 360 PRINT "SEU TEMPO DE REACAO
VARIOU ",INT (V*20/A);"%."
 370 PRINT
 380 PRINT "VOCE QUER TENTAR NOV
AMENTE?(S/N)"
 390 LET R$=INKEY$
 400 IF R$="S" THEN RUN
 410 IF R$<>"N" THEN GOTO 390
```

TK 90X

```
  10 REM ▚▞ PARA O TK90X ▚▞
  20 INK 7: PAPER 0: BORDER 0
  30 LET M=0: LET A=0: LET V=0:
DIM R(5): CLS
  40 FOR G=1 TO 5
  50 FOR P=1 TO RND*150+50
  60 PLOT INT (RND*250),INT (RND
*170)
  70 IF INKEY$<>"" THEN GOTO 90
  80 SOUND .1,-3: GOTO 70
  90 NEXT P
 100 FOR A=1 TO 4
 110 PRINT AT RND*20,RND*30;"■"
 120 NEXT A
 130 SOUND .05,15: LET C=0
 140 LET C=C+1: LET A$=INKEY$
 150 IF A$<>"" THEN GOTO 140
 160 LET R(G)=C/66+.05
 170 PRINT AT 19,0;"VOCE DEMOROU
 ";INT (R(G)*100)/100;"SEGUNDOS
PARA PARAR OS MOTORES."
 180 LET M=M+R(G)
 190 FOR J=1 TO 300: NEXT J
 200 NEXT G
 210 LET A=M/5: CLS
 220 PRINT "APOS 5 TENTATIVAS, S
UA MEDIA DE TEMPO DE REACAO FOI
DE ";INT (A*100)/100,"SEGUNDOS."
 230 FOR G=1 TO 5: LET V=V+ABS (
R(G)-A)
 240 NEXT G
 250 PRINT : PRINT "SEU TEMPO DE
 REACAO VARIOU ",INT (V*20/A);"%
."
 260 PRINT : PRINT "VOCE QUER TE
NTAR NOVAMENTE?(S/N)"
 270 LET R$=INKEY$: IF R$="s" TH
EN GOTO 30
 280 IF R$<>"n" THEN GOTO 270
 290 INK 0: PAPER 7: BORDER 7: C
LS
```

Obs.: Este é o caractere usado na linha 110 .



O MELHOR DE DOIS MUNDOS

Um televisor combinado com monitor próprio para micros permite alta qualidade de imagem e o uso de recursos sonoros, além de receber transmissões normais de tevê.

Talvez você possua um televisor-monitor sem ter se dado conta disso, pois muitos dos novos aparelhos receptores de tevê vêm equipados com saídas para videocassete adequadas também a micros.

Os problemas associados ao uso de televisores comuns como vídeos de computadores decorrem da maneira como eles recebem os sinais. As emissoras transmitem seus programas sob a forma de ondas de rádio, que são captadas pela antena e convertidas em imagens.

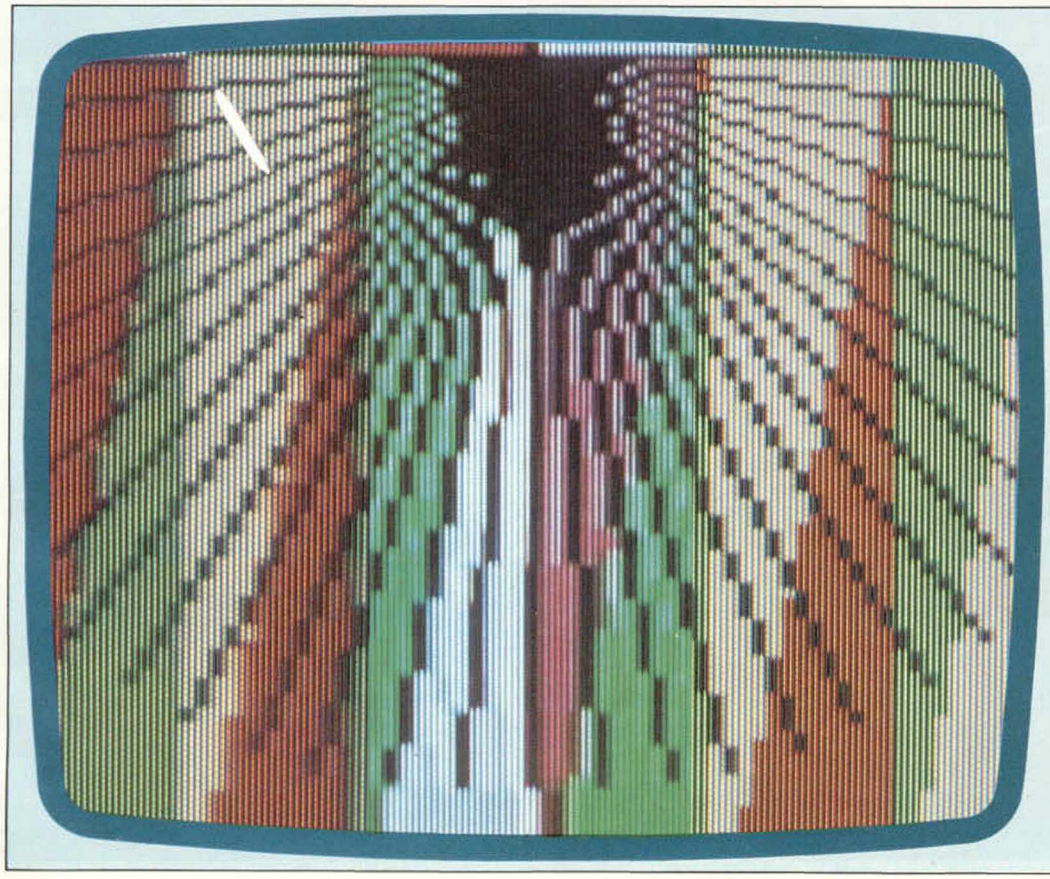
Um micro pode simular esse procedimento enviando, através de um modulador, sinais que depois, dentro do receptor, serão de novo alterados. Sofrendo duas modificações — no modulador e no receptor de televisão —, o sinal perde definição. O monitor, por sua vez, não necessita de modulação, recebendo o sinal da própria imagem, exibindo-a com nitidez.

Há televisores adaptados para se comportar como monitores e outros já projetados com entrada para videocassete. Estes se mostram mais eficientes, de vez que os primeiros são adequados a valores de tolerância bem mais amplos que os desenvolvidos para funcionarem como monitores.

Os televisores-monitores têm entrada de vídeo composto (indicada como "vídeo" ou "audio-visual"). Os diagramas e manuais que acompanham o aparelho mostram como ligar a uma dessas saídas os micros compatíveis. Em seguida, deve-se sintonizar o televisor do mesmo modo como se selecionaria um canal de emissora.

A principal vantagem do televisor-monitor está no recurso sonoro, bem superior ao de um monitor padrão. A maior parte dos microcomputadores — sobretudo os modelos da Microdigital e da Prológica — conta com televisores para a produção de efeitos sonoros. Um monitor padrão não possui recursos de som, ao passo que os televisores-monitores se valem de alto-falantes e amplificadores embutidos.

O desempenho superior do combinado televisor-monitor acabará fazendo desses aparelhos os mais procurados pelos usuários de micros.



Sinais e imagens

Existem três tipos principais de sinais produzidos por computadores. Todos os micros produzem sinais compatíveis com os de televisores, mas isso freqüentemente significa imagem insatisfatória. A maioria dos computadores também produz sinais para monitores, que fornecem melhor imagem mas são muito caros.

Os televisores-monitores combinam a qualidade do monitor com a capacidade de receber imagens de televisão. A diferença fundamental na qualidade entre televisores e monitores está no tipo de sinal com que trabalham. Estas três imagens foram produzidas no mesmo televisor-monitor, com três tipos diferentes de sinal: de tevê, de vídeo composto e de RGB (Red, Green, Blue).

O sinal de televisão *(no centro da tela)* fornece a imagem menos satisfatória; o de vídeo composto *(à esquerda)* é um pouco melhor; e o sinal RGB *(à direita)* dá os melhores resultados.

Maior nitidez
O televisor-monitor permite que seus videojogos sejam vistos com nitidez muito superior à conseguida num televisor comum.

Alta definição
A boa definição, na tela, de softwares gráficos, textos e desenhos torna menos cansativo o trabalho com programas extensos.

ORGANIZAÇÃO DE DADOS

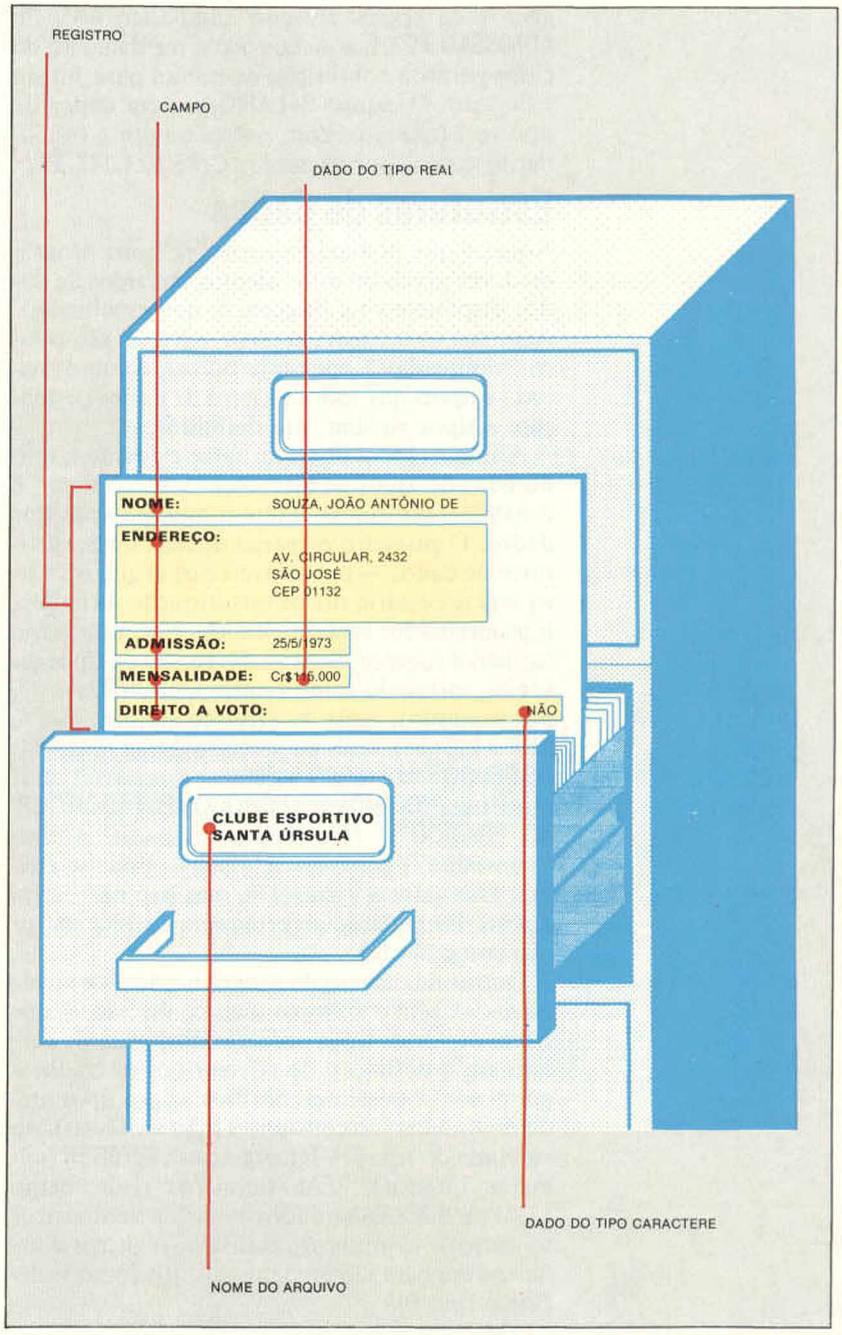
Quando bem elaborado, um banco de dados permite a organização e manipulação de informações, facilitando o acesso a todos os itens armazenados. Vamos ver como funciona esse recurso indispensável.

Muitas pessoas e instituições vêem-se obrigadas atualmente a manipular consideráveis volumes de informações estruturadas — secretários de clubes, bibliotecários, órgãos governamentais, corretores de imóveis, hospitais, escolas, empresas etc. E a tarefa de organização e atualização dessas informações fica extremamente facilitada pelo emprego de computadores e programas gerenciadores de bancos de dados. Antes, porém, de examinarmos as técnicas desenvolvidas para a utilização desses programas, vamos a uma breve revisão dos conceitos fundamentais envolvidos.

Para os programadores, os dados dividem-se em estruturados e não estruturados. Estes últimos são unidades isoladas, em geral armazenadas em variáveis ou em constantes — por exemplo, TENTATIVAS := TENTATIVAS + 1 ou IF CERTO = 1 THEN PROCEDURE INCREMENTO. Nos dois exemplos, TENTATIVAS e CERTO são variáveis; no primeiro caso, o valor da variável está sendo alterado e, no segundo, está sendo testado (com possível desvio para uma sub-rotina). As variáveis TENTATIVAS e CERTO somente podem assumir um valor de cada vez.

Os programadores que utilizam o BASIC acostumaram-se a pensar nos dados apenas como não estruturados — ou seja, como variáveis —, ao contrário do que ocorre no cotidiano, onde, na maioria das vezes, agimos a partir de dados estruturados. Considere, por exemplo, seu grupo de amigos: certamente você tem uma "estrutura de dados" própria para manipular o conjunto de dados referente a eles. Afinal, conhece seu nome, idade, sexo e outros dados como profissão, salário, hábitos, endereço etc. Um conjunto de fatos inter-relacionados sobre uma pessoa, conceito ou objeto recebe o nome de "registro". Cada registro compõe-se de um ou mais campos (de tipos iguais ou diferentes), e o conjunto de todos os registros é um arquivo.

Um exemplo ajudará a entender isso melhor: imagine o secretário de um clube esportivo com 1.500 associados. Para manter um controle eficiente, ele utiliza um fichário — com fichas de todos os sócios. Cada uma contém informações (dados) com características (tipos) diversas, relativas a um único associado. As informações incluem, por exemplo, sobrenome, nome, sexo, idade, ano de admissão, mensalidades pagas, salário, número do sócio, e assim por diante.



Os dados aceitáveis para os campos SOBRENOME e NOME são necessariamente representados por cadeias de caracteres alfanuméricos — afinal, 1374662 não seria apropriado para o nome de uma pessoa. O item SEXO aceita um dado de

tipo booleano, que admite apenas um em dois valores binários. IDADE será um dado de tipo integer (ou seja, um número inteiro, visto que nenhum secretário necessita saber que determinado sócio tem 37,624 anos de idade). Os dados para o campo ANO DE ADMISSAO também serão valores integer, mas pertencentes a uma faixa limitada. Assim, ANO DE ADMISSAO = 1530 não seria adequado, mesmo que o arquivo contivesse registros de sócios antigos; tampouco ANO DE ADMISSAO = 2001, a menos que o regulamento do clube permita a inscrição de nomes para futura admissão. O campo SALARIO contém dados de tipo real (números com partes inteira e fracionária) como, por exemplo, Cr$5.823.342,37.

Estruturas de dados

Aqueles que pretendem tornar-se bons programadores precisam estar atentos aos tipos de dados disponíveis na linguagem que escolheram. Algumas — inclusive o BASIC e o C — são pouco "tipificadas", enquanto outras, como o PASCAL, exigem que todos os itens de dados pertençam a tipos rigidamente definidos.

A importância disso torna-se evidente quando consideramos as diferenças entre o BASIC e o PASCAL no que se refere à manipulação dos dados. O primeiro emprega apenas duas estruturas de dados — as variáveis e os arquivos. Caso seja necessário maior estruturação de dados, o programador terá de estabelecê-la, pois o BASIC não a fornece. Suas variáveis são de tipos diversos, incluindo valores inteiros (PONTOS% = 7, por exemplo), reais de precisão simples (SALARIO! = 1.234,56), reais de precisão dupla (SALARIO DE PEDRO = 123.456,66666666666) e variáveis alfanuméricas (OTIMO$ = "MICROCOMPUTADOR CURSO PRATICO"). Não existe no BASIC o tipo "constante", como PI = 3,141592 — pode-se atribuir esse valor à variável PI, mas isso não exclui a posterior inclusão no programa de uma instrução como PI = 6,2.

Outras linguagens de programação — e talvez o PASCAL seja o exemplo mais conhecido — dispõem de vários tipos predefinidos, além de permitirem a definição de novos tipos de dados a partir dos já existentes. No PASCAL, os tipos predeterminados são constantes (não variáveis), ou variáveis do tipo CHAR (caracteres), INTEGER (números inteiros), REAL (números reais, como 12,71) e BOOLEAN (dados binários verdadeiros ou falsos). Conjuntos, matrizes, registros e arquivos também são predefinidos. Eis como se define o tipo DIA:

```
TYPE
  DIA = (SEGUNDA, TERCA, QUARTA, QUINTA,
         SEXTA, SABADO, DOMINGO);
```

Qualquer variável do tipo DIA usada no programa assume somente um dos valores especificados; portanto, se definíssemos FOLGA como uma variável pertencente ao tipo DIA, então FOLGA := NOVEMBRO não seria válido, ao contrário de FOLGA := DOMINGO.

Denominam-se bancos de dados os conjuntos de informações inter-relacionadas e estruturadas. Por convenção, os dados são organizados num arquivo (identificado por um nome), o qual compõe-se de registros (indicados por números), que, por sua vez, contêm um ou mais campos. Estes, finalmente, podem incluir um ou mais tipos de dados.

Supondo que o banco de dados do clube possua a mesma estrutura de um fichário, o mais provável é que seu principal código de acesso esteja baseado na ordenação alfabética dos sobrenomes dos sócios. Quando há necessidade de encontrar os sócios de sobrenome Souza, o secretário do clube percorre o fichário até encontrar uma ficha com a letra S. Em seguida, diminui a velocidade de busca e passa a procurar os sobrenomes que comecem por SO, até achar os registros dos Souzas. No entanto, caso ele precisasse verificar quantas sócias existem no clube, a pesquisa seria certamente muito mais trabalhosa. E ainda mais difícil e demorada seria a tarefa de descobrir quantos sócios recebem salários acima de 10 milhões de cruzeiros, têm idade superior a 37 anos e mais de dois filhos.

Esse tipo de tarefa é precisamente o que um programa ou sistema gerenciador de banco de dados realiza em pouco tempo ao simples toque de algumas teclas do computador. Segundo uma definição rigorosa, o banco de dados corresponde às informações, enquanto o programa que possibilita a introdução e manipulação dessas informações pelo computador recebe o nome de "sistema gerenciador de banco de dados" ou, de modo abreviado, SGBD.

Sistemas gerenciadores

A capacidade dos SGBDs varia bastante: os mais simples não passam de agendas de endereço eletrônicas que acessam um registro ao receberem um parâmetro simples como NOME = ?. Os SGBDs mais avançados contam com suas próprias linguagens de programação incorporadas, manipulando as informações por métodos bastante complexos e permitindo, inclusive, a transferência de campos de dados para outros programas (como folhas de pagamento, planilhas e processadores de texto).

Em resumo, um gerenciador de bancos de dados eficiente permite que o usuário defina o tipo dos dados admitidos em cada campo e, portanto, irá rejeitar uma entrada como esta: NOME = 143326 ou ANO DE ADMISSAO = 1530. Além disso, ele sempre emitirá uma mensagem de erro quando o usuário tentar encontrar registros claramente absurdos do tipo IF NOT (MASCULINO OU FEMININO) AND SALARIO < 0. Ao contrário das linguagens de programação como o BASIC — e de modo análogo às do tipo PASCAL —, os SGBDs exigem uma "tipificação rígida" para que não seja possível a introdução de dados errôneos durante a criação ou modificação dos registros.

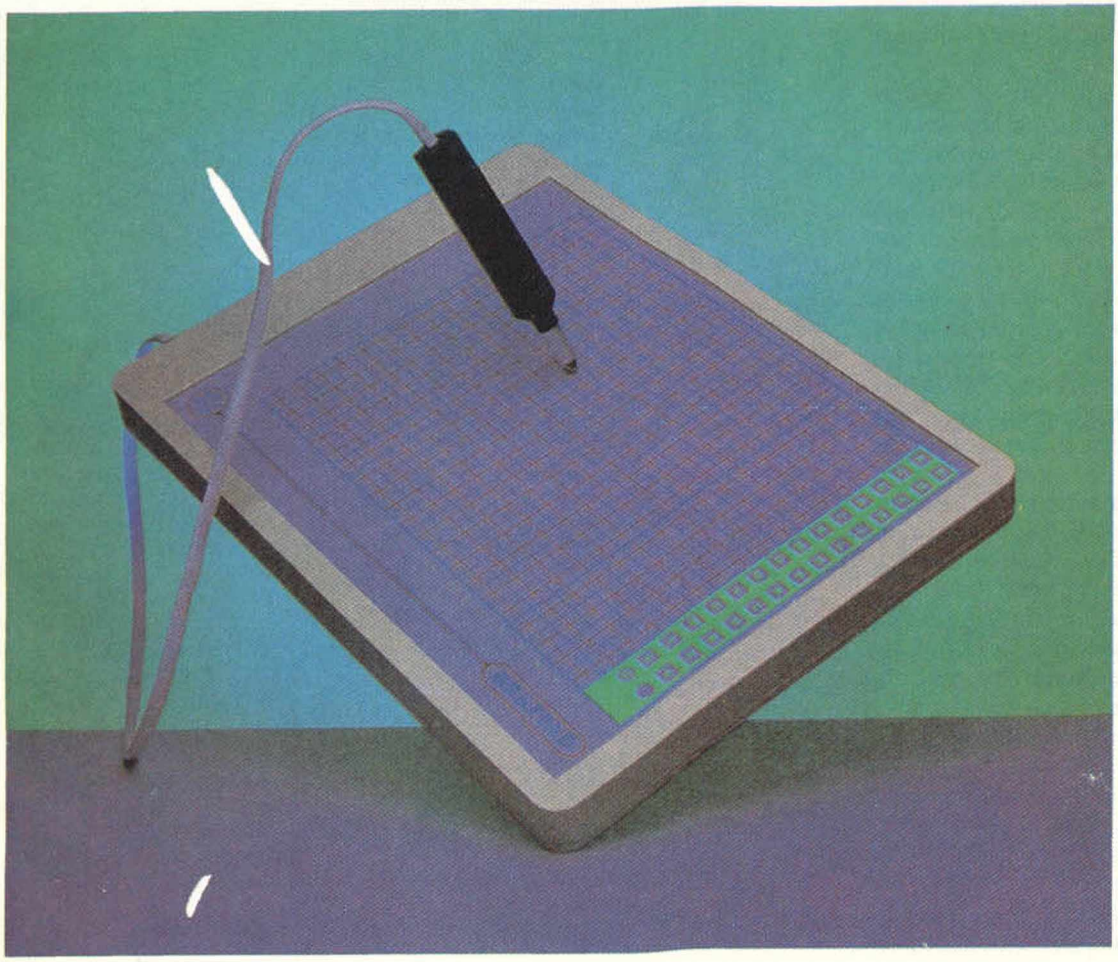
GRAFPAD

Compatível com o TK 90X, o Commodore 64 e o BBC Micro, o tablete digitalizador Grafpad, com seus três pacotes de software, é um equipamento ideal para a produção de desenhos e projetos em micros.

Tabletes digitalizadores estão entre os mais versáteis e úteis periféricos para microcomputadores. Têm importante aplicação como instrumento para desenho, arte a mão livre, projetos de circuitos eletrônicos, cartografia etc. E, além das aplicações diretas em desenho, constituem úteis dispositivos complementares de entrada. A máscara que os recobre pode mostrar todos os recursos disponíveis num programa, em palavras ou símbolos visuais. Tudo o que se tem a fazer é tocar o comando apropriado com a "caneta" que integra o equipamento, e o software processa a opção escolhida.

Tais sistemas eram utilizados em aplicações especializadas, vendidas sobretudo para projetistas e desenhistas. Mas os preços reduziram-se o suficiente para permitir a aquisição por usuários de micros domésticos. O Grafpad é um dos líderes entre os modelos de baixo custo, graças a suas boas características. Pode ser encontrado em versões específicas para o BBC Micro, o Commodore 64 e o Sinclair Spectrum (cujo compatível brasileiro é o TK 90X). A versão aqui ilustrada serve para o BBC.

O equipamento consiste em três elementos: o tablete, uma caneta e o software controlador. O tablete é conectado à saída do bus do micro e a caneta inserida lateralmente. A superfície do tablete apresenta-se dividida em 16 x 20 quadrados e mais uma coluna de comandos (um painel à direita com um conjunto de letras e números). Essa coluna permite o controle de partes do software, sem necessidade de se utilizar o teclado. Uma camada de plástico recobre e protege toda



Idéias gráficas
O Grafpad pode ser usado, com seu próprio software, para criar desenhos; ou, com os programas desenvolvidos pelo usuário, como periférico de entrada.

a superfície do tablete. É possível desenhar máscaras específicas com seus próprios comandos, e nelas traçar reticulados.

No interior do tablete existe um reticulado de 320 x 256 fios, distanciados 1,2 mm. A ponta da caneta é um contato minúsculo. Quando ela toca a cobertura de plástico do tablete, um chip ULA (Uncommitted Logic Array, "matriz lógica não comprometida") varre todos os fios, até detectar, por variação de capacitância, a posição da caneta. Esse rastreamento ocorre 2.000 vezes por segundo, garantindo a rápida localização do ponto tocado. Empunha-se a caneta pela faixa de metal que envolve sua ponta, a fim de ajudar o funcionamento seguro do sistema.

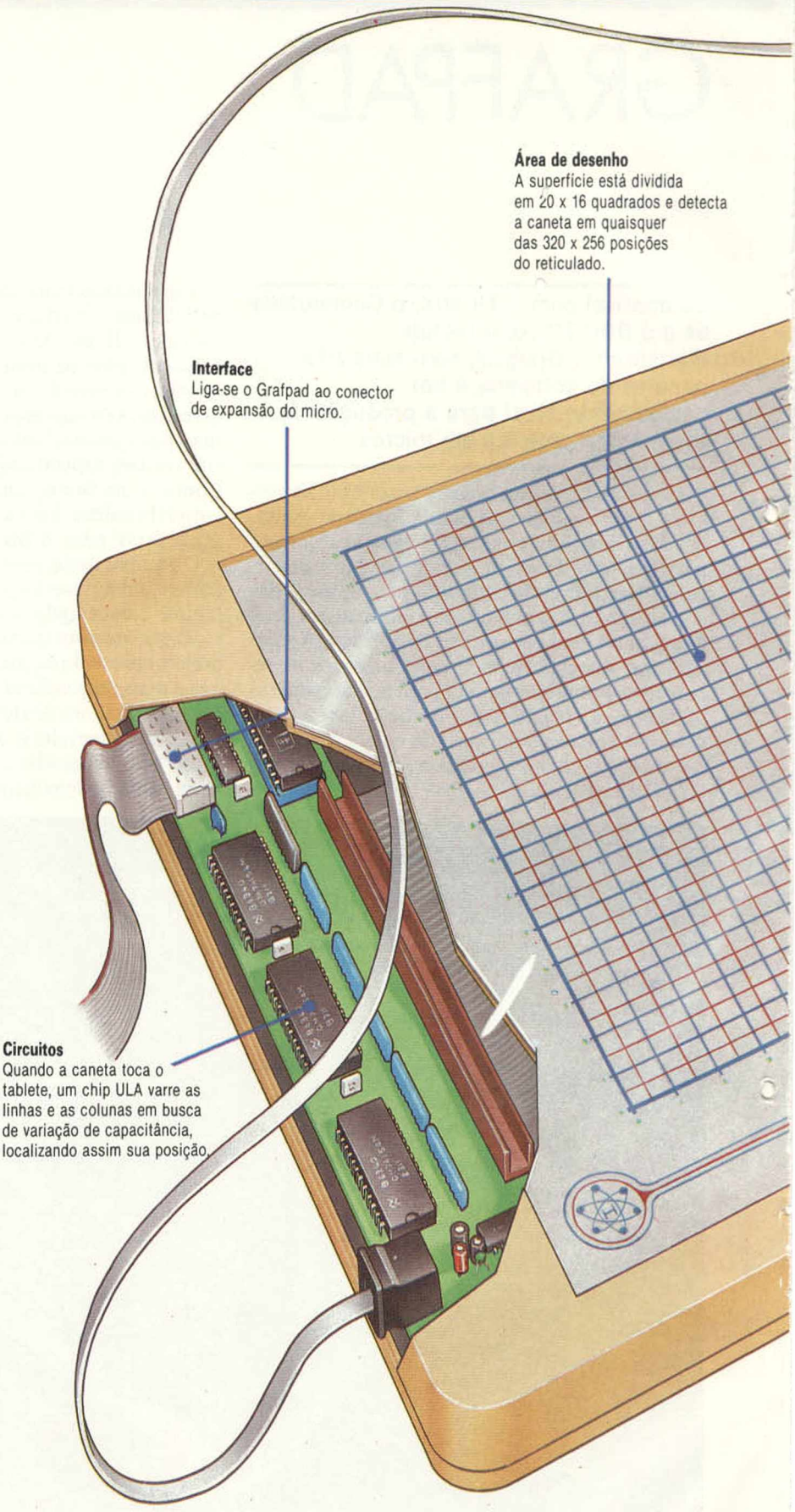
Quando a caneta encosta no tablete, o computador recebe a informação de suas coordenadas. O software determina o efeito exato que isso cria. Pode surgir um cursor em forma de cruz no vídeo, na posição correspondente, ou determinado comando ser acionado. E aqui a economia na construção do Grafpad começa a atrapalhar. Ele só detecta a caneta num reticulado de 320 x 256 posições, o que dificulta o desenho de detalhes muito reduzidos ou uniformes. E, também, o tablete é muito pequeno.

O Grafpad tem três pacotes de software, que vão de simples rotina de demonstração, via um programa de desenho, até um complexo pacote CAD (Computer-Aided Design, "desenho com auxílio de computador"). A rotina simples, tipo conheça-seu-tablete, pode ser incorporada aos programas do usuário (é fornecida em linguagem de máquina e em versão BASIC).

A rotina de desenho — um programa tipo faça-um-esboço — compara-se à maioria dos pacotes de arte disponíveis, mesmo àqueles que não usam tablete. E oferece todas as características básicas: retas, quadriláteros, círculos, triângulos e a mão livre, podendo preencher com cor áreas delimitadas. Faltam, porém, recursos mais sofisticados, como copiar e movimentar partes de um desenho. Nada há aqui que um software direcionado apenas pelo teclado não possa fazer, embora o Grafpad permita a criação de desenhos. A versão para o BBC apresenta apenas quatro cores de cada vez, com a desvantagem de respostas muito lentas.

O programa CAD é uma demonstração de alguns dos princípios envolvidos. Em primeiro lugar, criam-se os caracteres que serão usados nos desenhos. Para a eletrônica, por exemplo, essas formas seriam de componentes transistores, resistores etc. Também se podem criar portas lógicas, peças de mobiliário e mesmo padrões para azulejos. Transferem-se então esses caracteres para a prancheta, onde podem ser repetidos, arranjados e unidos a retas.

Isso é muito parecido com o funcionamento real de um pacote CAD. Mas o software do Grafpad não serve para uso profissional, pois faltam-lhe recursos para tanto. Por exemplo: capacidade de legendar diagramas, girar e graduar em escala os desenhos, ampliar determinada par-



SCOPUS

HÁ 10 ANOS DESENVOLVENDO TECNOLOGIA BRASILEIRA

No Brasil, a era da informática começou recentemente. Mas não para a SCOPUS que, desde 1975, trabalha em projetos, produções, criações e fabricações das mais avançadas, destacando-se no mercado porque aposta no talento e capacidade de técnicos, engenheiros e especialistas brasileiros. Não mede esforços. Sempre à procura de soluções modernas e criativas, desenvolve o que há de mais arrojado em matéria de informática. Sem copiar o que é feito fora. Conheça agora algumas das atuais conquistas tecnológicas da SCOPUS.

Tecnologia e sofistic

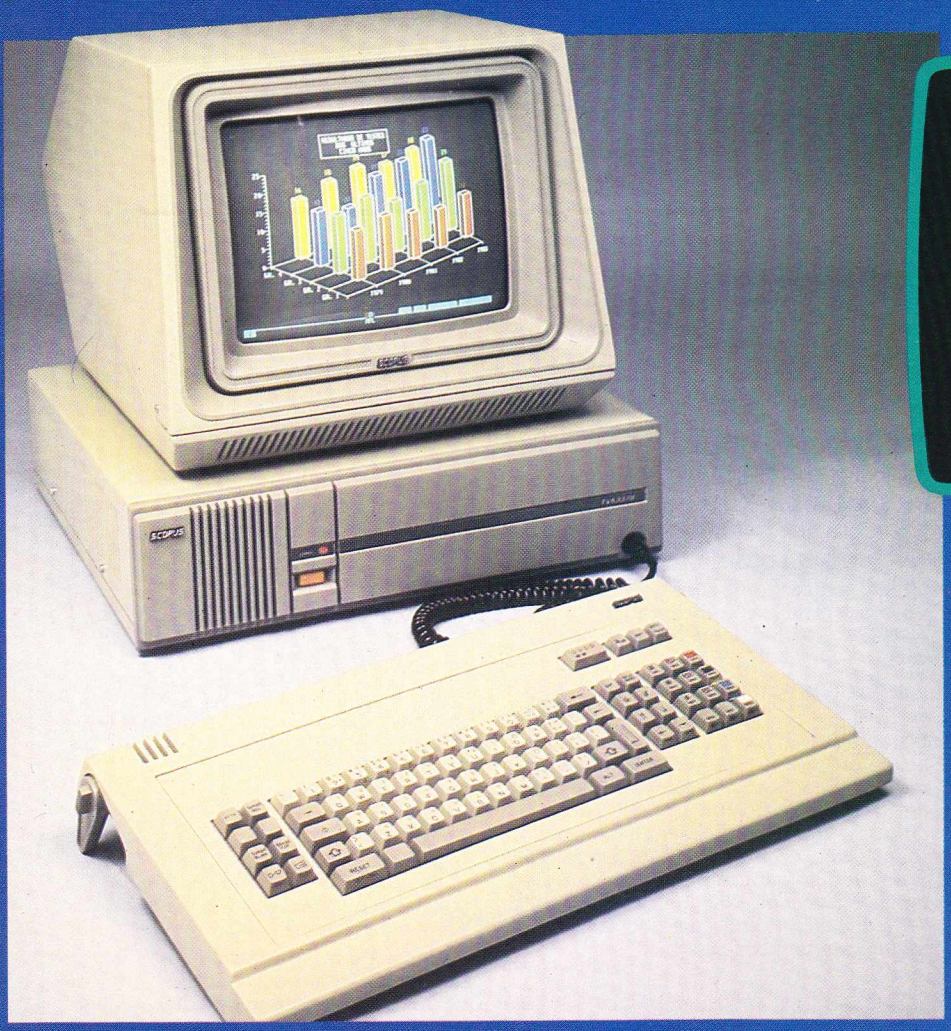
NEXUS 1684

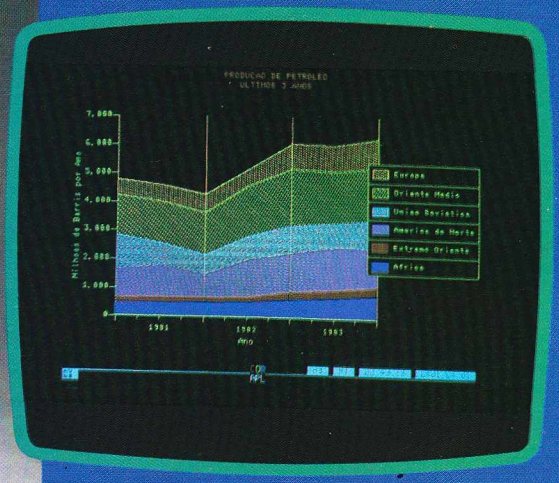
Como o mais recente integrante da família NEXUS 1600 de microcomputadores, o modelo 1684 traz uma importante novidade: a integração da unidade de disco magnético "Winchester", capaz de armazenar 10 milhões de caracteres (10 Mbytes). Dirigido a aplicações profissionais, o NEXUS pode ser utilizado para diversas operações, tais como análise financeira, análise de orçamento, balancete, entre outras tarefas gerenciais e administrativas. Ele se integra também a vários sistemas de grande porte, como computadores IBM, Burroughs ou Honeywell. Oferece características exclusivas, como o "clock" de CPU de 8 MHz e o tratamento de vídeo sem cintilação. Por trabalhar junto com a unidade Winchester, foi possível obter-se uma grande redução de custos além de um melhor aproveitamento do espaço físico.

TVA 3379

Com um hardware sofisticadíssimo e tecnologia (brasileira, é claro) das mais avançadas, o terminal de vídeo TVA 3379 tem como proposta principal facilitar a interpretação de dados fornecidos pelo computador através de gráficos a cores. É o único terminal desse tipo disponível para sistemas IBM existente no mercado. Usando o recurso das cores, o TVA 3379 facilita a leitura e inter-

ção em três versões.





pretação de qualquer informação, o que antes só poderia ser obtido através de tabelas numéricas. É a SCOPUS atenta a todo avanço e tendência do mercado mundial, facilitando a relação entre o homem e a máquina. Apenas usando o talento, o esforço e a criatividade de centenas de engenheiros brasileiros.

PC 2140

O mais novo e econômico "tradutor de linguagens", criado e desenvolvido com tecnologia totalmente nacional, o processador de comunicação PC 2140 permite a conexão simultânea de 32 equipamentos (terminais ou microcomputadores) com um computador de grande porte, através de uma única linha BSC-3. Isto permite o acesso ao banco de dados dos sistemas de grande porte, como IBM ou Burroughs, possibilitando ao usuário levantar importantes informações disponíveis no computador, entre elas, saldos de conta bancária, consultas ao pregão da Bolsa de Valores e verificação do estoque. O PC 2140 pode ser ligado a Unidades de Controle de Transmissão IBM 270X, 370X e ICA. A ligação dos equipamentos ao PC 2140 deve ser direta ou via modem, podendo-se também utilizar o modem de resposta automática.

Qualidade e assistência. A Scopus garante.

A avançada tecnologia do NEXUS 1600, PC 2140 e TVA 3379 já garante ao consumidor alto grau de qualidade e confiabilidade. Mas não é só isso que a SCOPUS oferece. Todos os serviços de manutenção são prestados por equipes de profissionais de alto nível técnico, em constante treinamento, que atendem rapidamente a qualquer chamado. Sempre investindo no aperfeiçoamento de sua estrutura de atendimento, a SCOPUS garante a mais elevada disponibilidade a seus equipamentos e a certeza de uma assistência técnica permanente.

DADOS TÉCNICOS

NEXUS 1600

UNIDADE DE SISTEMA

Placa de sistema: Microprocessador Intel 8088 (4,77 ou 8 MHz) • 64 Kbytes de memória EPROM • 256 Kbytes de memória RAM • 2 interfaces de comunicação serial: assíncrona e síncrona/assíncrona, padrão RS-232C, em velocidades até 19200 bauds • 5 conectores para placas de expansão.

Placa controladora de disco: controla duas unidades de disco flexível de 5 1/4", possui relógio-calendário de tempo real não volátil e duas interfaces para "joy-sticks".

Placa controladora de vídeo: modo alfanumérico: telas de 40x25 ou 80x25 caracteres, 16 cores • modo gráfico: telas de 320x200 pontos a 4 cores ou 640x200 pontos monocromáticas, interface para "light-pen".

Placa controladora de Winchester: controla a unidade de disco rígido tipo Winchester.

Unidade de disco flexível de 5 1/4": um acionador com cabeça dupla, densidade dupla, com capacidade de 320 Kbytes.

Unidade de disco rígido tipo Winchester: um acionador com capacidade de 10 Mbytes, slim.

MONITOR DE VÍDEO

Monocromático, 12", fósforo verde, interface padrão RGB, média resolução e configuração multinível, reproduzindo a informação da cor em tons de intensidades variáveis. Opcionalmente, pode-se configurar o Nexus 1600 com um monitor de vídeo cromático, 14", média resolução.

TECLADO

Tipo máquina de escrever, 86 teclas.

SOFTWARE BÁSICO

O Sistema Operacional SISNE, desenvolvido pela Scopus, emula o sistema MS-DOS 1.1 do microcomputador PC-IBM e contém todos os utilitários compatíveis com os utilitários do MS-DOS 1.1. Possui uma interface homem-máquina bastante poderosa e gerencia, com máxima eficiência, os recursos físicos do equipamento.

MÓDULOS DE EXPANSÃO

N1664: expansão de memória RAM (448 Kbytes) • **N1667**: co-processador CP-8087 • **N1669**: interface paralela para impressora.

MÓDULOS PARA COMUNICAÇÃO DE DADOS

Sistema Incox: interface coaxial para conexão a redes IBM 3270 • **NX-3270**: módulo para comunicação com sistemas IBM sob protocolo BSC-3 • **NX-830**: módulo para comunicação com sistemas Burroughs sob protocolo Poll-Select • **NX-VIP**: módulo para comunicação com sistemas Honeywell Bull sob protocolo VIP 7700 • **NX-Z**: módulo para comunicação assíncrona serial de dados.

TVA 3379

MÓDULOS

Unidade de vídeo: apresenta tela útil de 1920 ou 2560 posições, respectivamente para os modelos 2 ou 3. Possui duas linhas adicionais de informações no vídeo para mensagens do sistema e de erros, além de possibilitar a apresentação de cronômetro e indicador de linha e coluna. O monitor é cromático, de alta resolução. O cinescópio, de 14 polegadas, pode ser configurado para que se apague automaticamente, evitando a queima desnecessária do fósforo.

Unidade lógica: comporta as placas de controle do terminal, de geração de caracteres plano triplo, e as interfaces de comunicação com o vídeo, unidade controladora e teclado.

Teclado: comunica-se com a unidade lógica de forma serial e permite a conexão de um dispositivo auxiliar de entrada. Está disponível em quatro opções: APL Português, APL Inglês, "Typewriter" Português e "Typewriter" Inglês, todas com bloco destacado de teclas de cor e de símbolos programáveis.

PC 2140

CARACTERÍSTICAS FUNCIONAIS

Interface de comunicação com terminais ou micros: serial, assíncrona, código ASCII, full duplex, padrão elétrico RS-232C.

Placa interface de comunicação: cada placa permite que até 8 dispositivos sejam conectados ao PC 2140 com seleção de velocidades independentes.

Conexão de modem: a interface de comunicação com os terminais ou micros permite a ligação de modem, podendo ser constante ou por meio de linhas discadas, através de centrais telefônicas.

Interface de comunicação com o computador: serial, síncrona, sob o protocolo BSC-3, padrão elétrico RS-232C, em velocidades até 9600 bps. Emula as unidades de controle tipo IBM 3271 ou 3274 com 32 terminais IBM 3277 ou 3278, modelo 1.

Conversão de códigos: adotados pelos terminais (ASCII) em códigos adotados pelo computador (EBCDIC).

Execução de comandos: permitem ao computador ativar funções como, por exemplo, estabelecer conexões com os terminais, desfazer conexões, transmitir textos e informar o estado de ligações.

Verificação de paridade: em todas as mensagens com o computador, o PC 2140 gera e verifica os caracteres de controle de bloco (CRC-16), pedindo retransmissão automática em caso de erro.

MODELOS

O PC 2140 está disponível em quatro opções, que diferem apenas em relação ao número de placas de controle dos terminais.

PC 2140/8: controla até 8 terminais (1 placa).

PC 2140/16: controla até 16 terminais (2 placas)

PC 2140/24: controla até 24 terminais (3 placas)

PC 2140/32: controla até 32 terminais (4 placas)

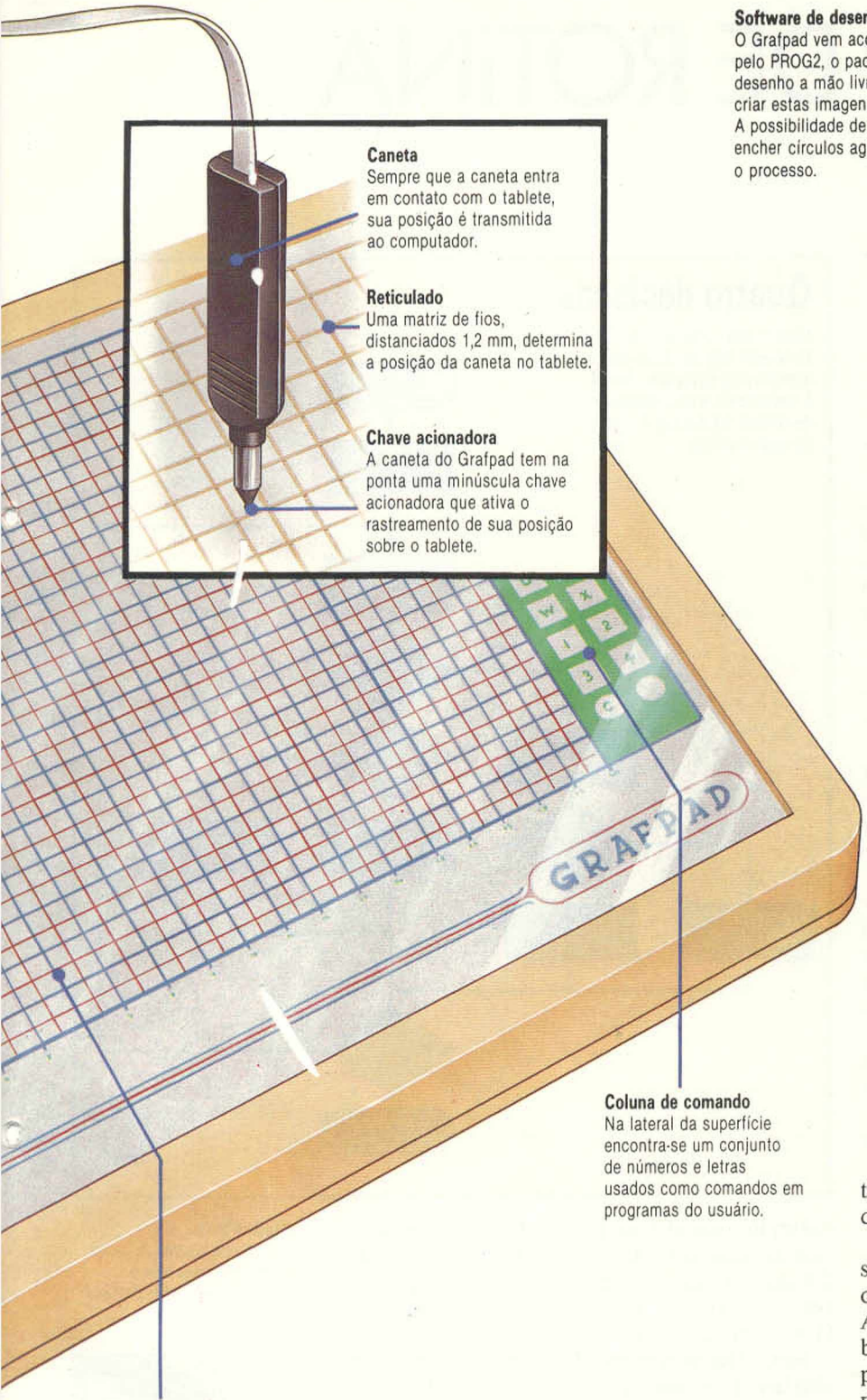
CENTROS DE ASSISTÊNCIA TÉCNICA

BELÉM: Tel.: (091) 228-0011 - ramal 242
BELO HORIZONTE: Tel.: (031) 224-8457
BRASÍLIA: Tels.: (061) 224-9856 e 224-9835
CAMPINAS: Tel.: (0192) 42-7333 - BIP: 657
CUIABÁ: Tels.: (065) 322-0542 321-5792 - BIP: 139
CURITIBA: Tel.: (041) 262-5211 - BIP: 150
FORTALEZA: Tel.: (085) 224-7999 - BIP: DA2
LONDRINA: Tel.: (0432) 22-0558
PORTO ALEGRE: Tel.: (0512) 32-1200 - BIP: 798
RECIFE: Tel.: (081) 221-3100 - BIP: 765
RIBEIRÃO PRETO: Tels.: (016) 636-3967 634-3331/3509 - BIP: 76B
RIO DE JANEIRO: Tels.: (021) 240-4925 e 240-6114
SALVADOR: Tel.: (071) 233-4117 - BIP: D12
SÃO JOSÉ DOS CAMPOS: Tels.: (0123) 22-8247 21-2004 - BIP: 536
SÃO PAULO: Tel.: (011) 255-7744

SCOPUS tecnologia s.a.

Filiada à ABICOMP

As especificações contidas neste folheto podem ser alteradas pelo fabricante sem aviso prévio.



Caneta
Sempre que a caneta entra em contato com o tablete, sua posição é transmitida ao computador.

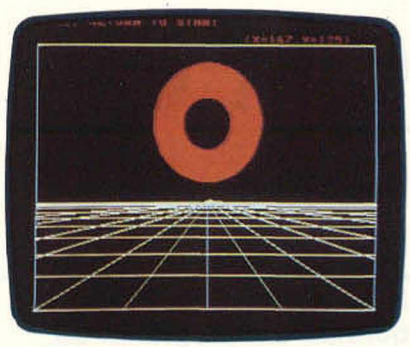
Reticulado
Uma matriz de fios, distanciados 1,2 mm, determina a posição da caneta no tablete.

Chave acionadora
A caneta do Grafpad tem na ponta uma minúscula chave acionadora que ativa o rastreamento de sua posição sobre o tablete.

Coluna de comando
Na lateral da superfície encontra-se um conjunto de números e letras usados como comandos em programas do usuário.

Cobertura
Um plástico transparente protege a superfície do equipamento, podendo também fixar máscaras para outras aplicações.

Software de desenho
O Grafpad vem acompanhado pelo PROG2, o pacote de desenho a mão livre usado para criar estas imagens.
A possibilidade de traçar e encher círculos agiliza bastante o processo.



te da tela, posicionar com precisão formas pequenas, e assim por diante.

Maior flexibilidade na correção de erros é essencial e, em geral, o programa CAD despreza o uso do Grafpad como periférico de entrada. Apesar da pequena coluna de comandos no tablete, muitos outros precisam ser introduzidos pelo teclado, e a operação, no todo, torna-se muito incômoda.

O Grafpad, em si, mostra-se um periférico versátil e, ainda que limitado quanto a área, resolução e confiabilidade — em função de seu baixo custo —, oferece bom desempenho. O software que acompanha o sistema, no entanto, desaponta — a unidade atrai mais aqueles que querem escrever seus próprios programas. Ainda assim, tabletes como este permitem a exploração de novas possibilidades e, por isso, devem proporcionar impulso considerável no campo de desenhos mais avançados para microcomputadores.

CHECK-UP DE ROTINA

Não há meios de testar, em tempo normal, um programa de complexidade razoável. Mesmo assim, vale a pena criar métodos seqüenciais de teste, a partir da estrutura piramidal do programa.

O BASIC ou qualquer outra linguagem interpretada permite testar as instruções quando são escritas. Pode-se digitar RUN a qualquer momento e ver o que acontece. Na maioria das máquinas, procurar erros resume-se a interromper com BREAK um programa em execução, obter com um PRINT os valores das principais variáveis, verificar esses valores e, então, CONTinuar. Embora erros mais óbvios sejam detectados e corrigidos, esse controle não substitui os testes aplicados quando o programa está acabado.

O teste de validade tem por objetivo garantir que um programa fará exatamente o que se espera dele. Para qualquer conjunto admissível de dados de entrada, ele deve produzir a saída correta; e, para qualquer entrada não permitida, deve executar as ações apropriadas. Aparentemente, pode-se fornecer ao programa uma amostra de todas as entradas admissíveis, mas isso seria inviável para a maioria dos programas. Um que servisse apenas para somar dois números inteiros e imprimir o resultado deveria ser testado para todos os valores possíveis — sem falar nos valores "ilegais", que também precisariam ser testados.

Outra possibilidade seria examinar todos os caminhos lógicos ao longo do programa. Pode-se encontrar cada um deles seguindo uma rota ao longo do diagrama de controle de fluxo (fluxograma), do começo até o fim. Cada ramificação de trajeto leva a caminhos alternativos, e cada loop acrescenta novas opções. A figura 1 mostra um programa simples, um loop contendo várias instruções IF-THEN. Há vários caminhos dentro do corpo do loop, que é executado dez vezes. São cerca de doze linhas de instrução, mas o número de caminhos diferentes chega a 1.398.100, o que descarta essa maneira de testar.

Não funciona nem o teste exaustivo dos dados nem o teste exaustivo da lógica; na verdade, não existem meios de testar um programa de certa complexidade num tempo razoável. Apesar disso, vale a pena criar métodos de teste. Uma suposição razoável é a chamada "equivalência de classe": se a máquina opera corretamente com um dado de determinado tipo, ela irá operar corretamente com todos os dados desse tipo. Por exemplo, quando um trecho de instruções verifica se uma entrada está na faixa de 1 a 100, devem-se testar entradas menores que 1, maiores que 100 e dentro da faixa esperada (1 = < valor = < 100).

Simplifica-se o exame dos caminhos lógicos assinalando os pontos de entrada de cada rotina (em princípio, deveria haver apenas uma entrada para cada rotina) e, dentro das rotinas, cobrindo todos os resultados possíveis de cada ramificação de decisão. Na figura 2, temos uma rotina para ajustar pontos de bônus num jogo que considera os parâmetros de entrada BONUS, NIVEL e ACERTOS. Ela pode ser escrita:

```
6030 IF NIVEL > 2 AND ACERTOS = 10 THEN
     BONUS = BONUS * NIVEL
6040 IF NIVEL = 6 OR BONUS > 2000 THEN
     BONUS = BONUS + 100
```

Quatro decisões

Mesmo uma construção simples como este loop não pode ser exaustivamente testada, devido à multiplicidade das condições de entrada: há mais de 1 milhão de rotas distintas.

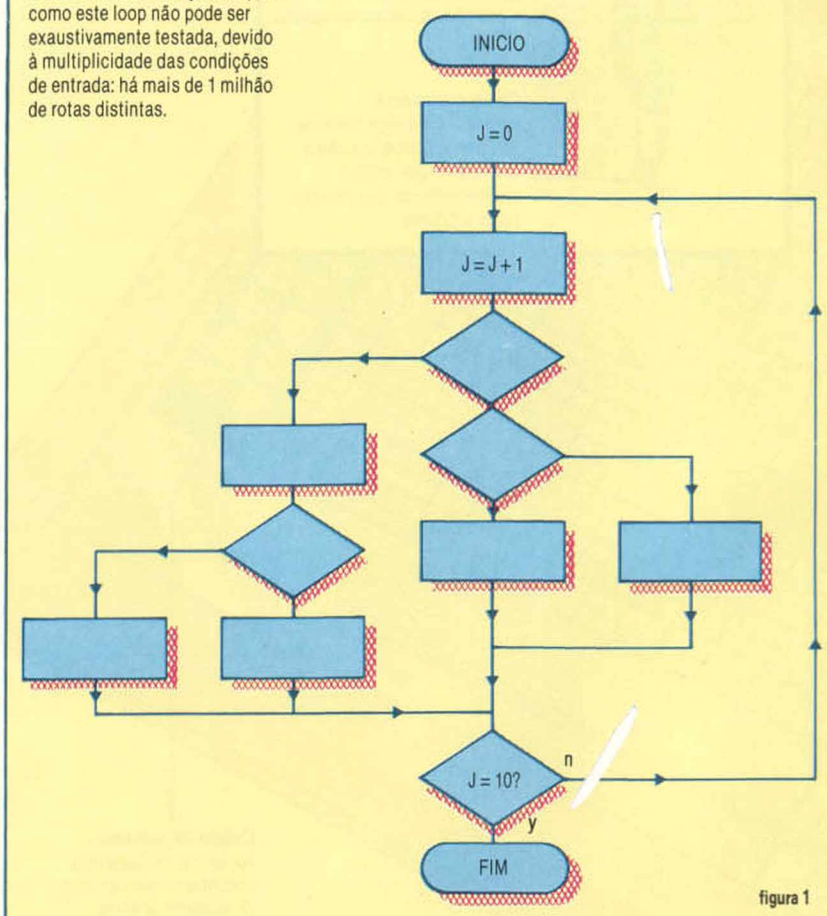


figura 1

Testando o teste
Um conjunto completo de dados de teste calculados para o exemplo ilustrado no fluxograma deveria ser semelhante ao seguinte:

ENTRADA			SAÍDA
NIVEL	ACERTOS	BONUS	BONUS
6	10	200	1300
4	10	550	2300
7	10	550	3950
4	10	200	800
7	10	200	1400
1	20	2500	2600
1	20	550	550
6	5	200	300
6	50	200	300
4	5	2500	2600
7	50	2500	2600
4	50	550	550
7	5	550	550

Para cobrir os resultados de cada expressão condicional, precisaríamos considerar as entradas de cada uma que causariam saída "sim" ou "não". Em ambas as decisões, olhamos os efeitos de duas variáveis combinadas por um operador lógico (E e OU). Ou seja, consideramos os valores combinados das variáveis, e não seus valores individuais. Para entender a razão disso, imaginemos o que aconteceria se testássemos os valores 4 e 1 para NIVEL, e 10, 5 e 20 para ACERTOS na primeira decisão. Quando NIVEL = 4, os três valores de ACERTOS seriam testados; mas não quando NIVEL = 1. Nesse caso, parte de uma decisão teria "mascarado" outra parte. Então, se pudermos testar cada parte separadamente, será melhor simplificar decisões compostas.

A figura 3 mostra que, com quatro decisões binárias, existem dezesseis (2^4) resultados possíveis; não precisamos cobrir todos eles. Um bom começo é tabelar as condições que produzem resultados sim ou não para cada decisão:

	1	2	3	4
sim	NIVEL > 2	ACERTOS = 10	NIVEL = 6	BONUS > 2000
não	NIVEL = 2 NIVEL < 2	ACERTOS < 10 ACERTOS > 10	NIVEL < 6 NIVEL > 6	BONUS = 2000 BONUS < 2000

A tabela pode então ser usada para derivar valores dos dados representativos do teste. Por exemplo, para o percurso a-d-f-i (figura 3), NIVEL precisa ser maior que 2 e igual a 6, ACERTOS deve ser diferente de 10 e BONUS pode ter qualquer valor porque não está envolvido nas decisões. Os valores NIVEL = 6, ACERTOS = 20 e BONUS = 150 seguiriam esse caminho — como fariam tantos outros. O percurso a-b-e-h-j poderia ser testado com NIVEL = 4, ACERTOS = 10 e BONUS = 600 (falamos do valor de *entrada* de BONUS, que mais tarde pode ser multiplicado pelo valor de NIVEL).

Devem-se calcular previamente os resultados a serem obtidos com cada conjunto de dados, para comparação aos produzidos pelo teste. Os dados de entrada meramente testarão se o programa "roda". Para testar se ele faz o que deve, a saída precisa ser calculada de antemão — como no exemplo *à esquerda,* para NIVEL, ACERTOS e BONUS.

Equipados com um método para "exercitar" nosso software, precisamos agora de um método para abordar programas de maior extensão. Nesse ponto, percebe-se outra vantagem da programação estruturada: programas escritos como uma coleção de módulos independentes dispostos numa hierarquia permitem testar cada módulo individualmente. Podemos começar com o módulo prioritário (o do topo da hierarquia) e "descer" a pirâmide. Só testamos um módulo quando todos acima dele tiverem sido testados, e usamos módulos já testados para fornecer dados aos inferiores.

A menos que seja o primeiro, o módulo a ser testado terá acima dele um módulo gerenciador

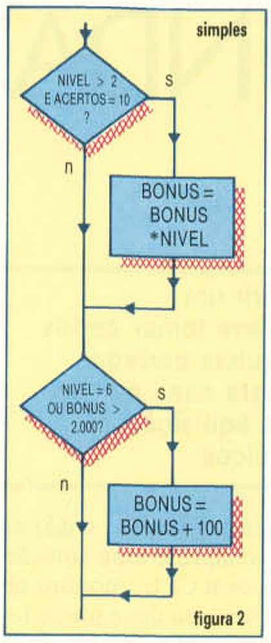


figura 2

Decompondo a decisão
Nomeie as ligações e simplifique decisões compostas; isso facilita os testes.

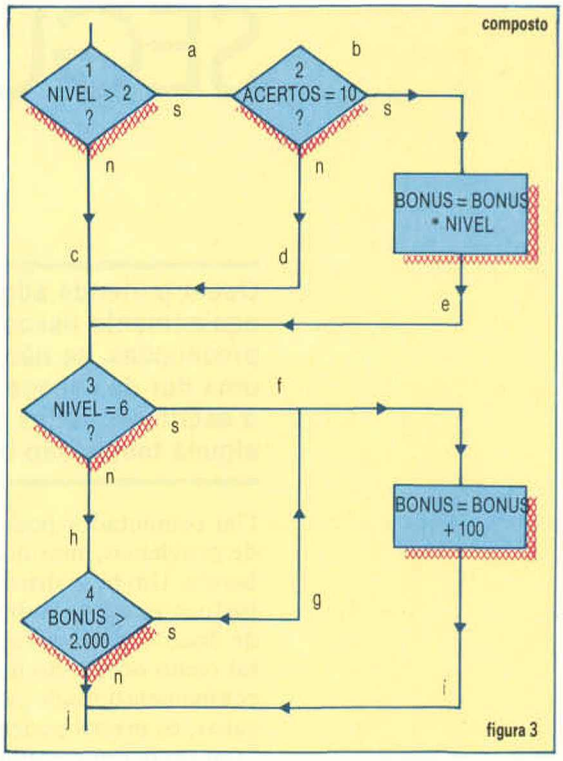


figura 3

plenamente testado. Os inferiores a ele, que poderíamos chamar de "elementares", ainda não foram testados, não sendo confiáveis.

Simulam-se os elementares por curtos trechos de programa que simplesmente produzem dados apropriados quando utilizados pelo módulo que está sendo testado. Às vezes, chama-se esse arranjo de "couraça de teste" — uma armação em que rotinas modulares podem ser colocadas para teste. A figura 4 corresponde a um esquema desse princípio. Os módulos 1, 2 e 3 foram testados, o módulo 4 está em teste e 5, 6 e 7 são simulados.

Os testes, constituindo importante parte do ciclo de vida de um programa, precisam ser bem documentados. Vale a pena manter registros dos dados de teste derivados de uma rotina. Se surgirem problemas, não haverá necessidade de repeti-los, podendo-se verificar, também, por que razão se mostraram inadequados.

Teste do topo à base
Simplifica-se o teste pela abordagem do topo à base, uma vez que cada módulo é testado como se escreve, tanto isolado quanto associado a outros já testados.
O comportamento de módulos ainda não escritos pode ser simulado criando-se rotinas que produzem, em suas saídas, dados gerados artificialmente, mas coerentes com os previstos.

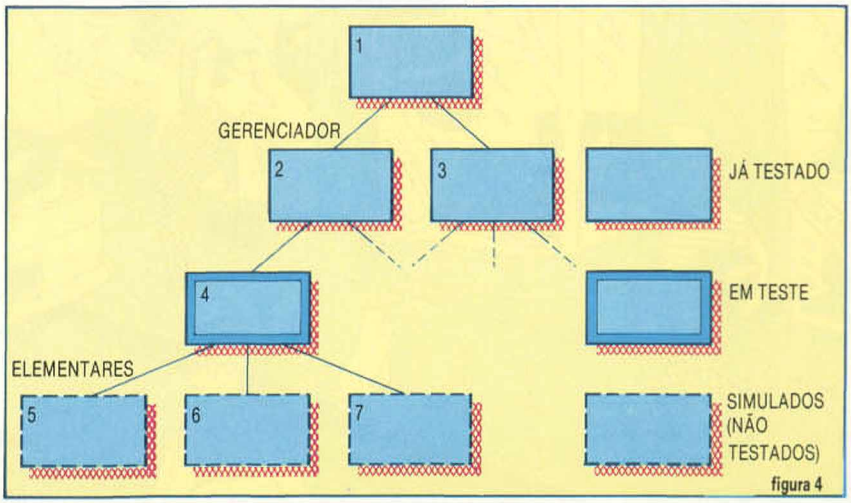


figura 4

SEGUNDA MÃO

Quem pretende adquirir um equipamento usado deve tomar certas precauções, se não quiser carregar uma dor de cabeça para casa ou para o escritório. Antes da aquisição, alguns testes são básicos.

Um computador pode resolver dúzias e dúzias de problemas, mas nem sempre é uma solução barata. Um bom sistema, com CPU, monitor de fósforo verde ou âmbar, uma ou duas unidades de disco e impressora, custa caro. No entanto, tal como ocorre no mercado de automóveis, o equipamento usado sai mais barato. Em certos casos, os preços podem chegar à metade do cobrado por um computador novo.

Mas você precisa saber o que está comprando. Antes de mais nada, é necessário determinar exatamente para que o micro está sendo comprado — jogos, contabilidade, processamento de texto, cálculos financeiros, folhas de pagamento, faturamento, simulação, controle de processos etc.

Em teoria, qualquer equipamento pode fazer tudo isso (uma coisa de cada vez, no mínimo), mas não compre nada até ter certeza de que existe software apropriado para as tarefas que o micro vai executar.

Nessa etapa, a situação é a mesma de quem compra um micro novo. Convém conhecer o software, experimentá-lo, verificar se ele atende às necessidades da tarefa a ser executada e só então decidir que combinação máquina/programa deve ser escolhida.

Contudo, tal decisão precisa levar em conta outros fatores, como, por exemplo, a adaptabilidade da máquina. Procure saber:

- Se ela tem até 32 Kbytes de memória, há no mercado expansão para 64 Kbytes? (Os programas mais eficientes requerem esta quantidade.)
- A configuração mínima já inclui alguma interface para periféricos?
- Há no mercado interfaces para as aplicações desejadas?
- É grande a variedade de software para essa máquina?
- Quanto custa fazê-la comunicar-se com um

serviço público de transmissão de dados como o Videotexto?

Depois de avaliados esses aspectos, pode-se decidir com segurança pelo menos o tipo de máquina que executará o trabalho pretendido.

A etapa seguinte consiste em procurar o equipamento, e aí o caminho toma duas direções: pode-se buscá-lo nas lojas ou nos classificados de grandes jornais, onde pessoas, pelas mais diversas razões, anunciam equipamentos usados para vender. O preço é questão muito importante, mas não a única. Exemplo: um micro usado vendido numa loja pode ser mais caro do que o comprado de um particular; em contrapartida, a loja em geral oferece alguma garantia (trinta dias, em média), e sempre se pode conseguir software grátis.

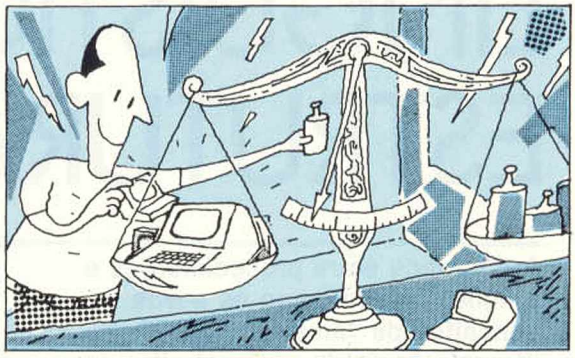
Chega, então, o momento de avaliar a máquina para saber se ela está em boas condições. A primeira coisa que se faz é pedir ao dono que ligue o micro — isso pode evitar uma porção de surpresas. Se o equipamento emprega fitas cassete, começa-se o teste carregando um programa na memória do micro. Se não houver, melhor ainda: digita-se um, para testar a gravação e o carregamento. Ele pode ser curto, como o que segue, para equipamentos compatíveis com o TRS-80, facilmente adaptável para outras linhas de micros:

```
10 CLS
20 FOR I = 0 TO 255
30 PRINT (0 + 1), CHR$(I);
40 NEXT I
50 GOTO 50
```

Isso vai exibir todos os caracteres disponíveis em BASIC e servirá também como teste inicial do teclado. Deve-se carregar e rodar o programa mais de uma vez, para ter certeza de que a lógica do micro está em ordem. Por fim, verifica-se o tamanho da memória com um comando do tipo PRINT MEM. Como esse comando pode variar de um BASIC para outro, consulte antes o manual da linguagem. Um micro com 64 Kbytes de RAM deve responder com pelo menos 48 Kbytes livres; se estiver com problemas, poderá dar uma resposta absurda como, por exemplo, 722 bytes livres.

O teclado constitui outro item a ser testado cuidadosamente. O ideal é digitar uma série de instruções, um texto ou experimentar todas as teclas, verificando se elas não ficam presas. E, em caso afirmativo, se repetem caracteres, defeito que às vezes compromete o teclado inteiro. Depois de todos esses testes, passa-se à avaliação do monitor de vídeo. Em dez minutos de funcionamento, ele evidenciará qualquer problema que porventura tenha.

Esse periférico não deve apresentar nenhum ruído (no máximo zumbido de bobinas); os caracteres que aparecem no alto e no pé da tela devem ter o mesmo tamanho; e a imagem precisa



estar centrada — aspecto que se verifica com este programa:

```
10 CLS
20 FOR I = 0 TO 1023
30 POKE (15360 I),191
40 NEXT
50 GOTO 50
```

Se o quadro gerado se apresenta descentrado, bastarão alguns ajustes. É o máximo de problemas que se pode aceitar de um monitor.

Para teste da impressora, usa-se o mesmo programa gerador de caracteres na tela, empregando-se LPRINT no lugar de PRINT. Será suficiente para verificar se todas as trilhas de comunicação micro—interface—impressora encontram-se perfeitas, e também se a própria impressora funciona bem.

No caso das unidades de disco, há dispositivos que apresentam problemas com alguma freqüência, por causa de sujeira. Assim, podem não estar funcionando bem apenas por estarem sujos. Um bom teste é colocar o sistema operacional e executar alguns programas gravados em linguagem de máquina (como o próprio interpretador BASIC) e formatar um ou dois disquetes. Se o sistema estiver em ordem, a formatação deverá se concluir com êxito; havendo problema, tenta-se de novo, com um disquete virgem. Se ainda ocorrerem erros durante a formatação, será melhor procurar outro equipamento.

LINGÜÍSTICA ESTRUTURAL

A diferença entre procedimentos e funções, assim como os novos conceitos de "âmbito" e "transferência de parâmetro", ajudará a estruturar de modo mais eficiente os programas em PASCAL.

Finalizando a investigação das estruturas de dados no PASCAL, examinaremos alguns métodos de estruturação de programas. Nesta fase, definiremos novos procedimentos e funções que complementam os fornecidos pela linguagem, como o procedimento WRITE e a função SQRT. Contudo, antes disso, será necessário retomar a distinção entre procedimento e função. Ao dizermos, por exemplo,

```
WRITE ('ALO!')
```

esperamos que o string de caracteres fornecido como parâmetro único apareça no arquivo de saída padrão. Ou seja, WRITE identifica um subprograma responsável pelo envio dos dados, sob a forma de um fluxo de caracteres, para o dispositivo de saída. Mas qual seria o resultado da seguinte "instrução"?

```
SQRT (256)
```

Nenhum, além de gerar uma mensagem de erro, visto não ser uma instrução legítima. Por outro lado,

```
WRITE (SQRT (256))
```

não apenas é válida, como também apresenta na tela algo do tipo 1,60000E+01. A instrução WRITE identifica um processo que envia dados para a saída, sendo um procedimento designado por seu próprio nome. Um outro procedimento, PAGE (F), serve para iniciar uma nova página no arquivo de texto F.

No caso do identificador SQRT, sempre atribuímos a ele um único "argumento" numérico para que forneça um resultado — a raiz quadrada do argumento. Desse modo, a primeira distinção entre procedimentos e funções é a de que não existe um comando função. Assim como 16 isoladamente nada significa (trata-se apenas de um valor), expressões isoladas como SQRT (X/3) ou ODD (N) também não possuem nenhum significado. Os identificadores de função comportam-se como variáveis não inicializadas, que são calculadas toda vez que seus nomes surgem numa instrução. Calcula-se o valor único resultante a partir do(s) valor(es) do(s) argumento(s) relativo(s) à função — ou seja, o resultado é uma função do(s) argumento(s).

Os procedimentos, ao contrário, não fornecem valores e, portanto, não podem ser usados em expressões. A instrução

```
N := WRITELN
```

é claramente ilegítima, do mesmo modo que LET N = PRINT o seria em linguagem BASIC.

A elaboração de programas extensos fica bastante facilitada, em PASCAL, pela possibilidade de escolhermos os nomes de procedimentos e funções, bem como de controlarmos o acesso a eles e sua vinculação mútua. Considere o seguinte programa:

```
PROGRAM INCOMPLETO (INPUT, OUTPUT,
  ARQUIVO); {DECLARACOES ..}
BEGIN
  OPEN (ARQUIVO);
  WHILE NOT EOF (ARQUIVO) DO
    BEGIN
      READ (ARQUIVO,ITEM);
      PROCESSE (ITEM);
    END;
END.
```

Embora você ainda não saiba como manipular arquivos, não será difícil compreender o que faz esse programa. Na parte da declaração há um procedimento para localizar um arquivo e abri-lo para leitura (OPEN), e um outro para manipular cada item de dados (PROCESSE).

O programa também utiliza dois identificadores predefinidos do PASCAL. O procedimento READ (utilizado até agora para ler entradas de texto) serve para a leitura de quaisquer dados, estruturados ou não, provenientes de um arquivo.

A função booleana EOF (End of File, "fim de arquivo") fornece um valor "verdadeiro" quando o último registro do arquivo é lido. Para arquivos de texto, a função EOLN (End of Line, "fim de linha") resultará "verdadeira" quando o último caractere da linha for lido.

A diferença fundamental entre programas, procedimentos e funções é pequena — afinal, todos são módulos com suas próprias descrições de dados locais e seus núcleos de instruções. A única variação sintática diz respeito ao cabeçalho, um programa assim composto: pela palavra reservada PROGRAM; por um identificador, escolhido pelo usuário, que designa o nome do programa; e por uma lista de parâmetros que indica os arquivos utilizados pelo programa.

No cabeçalho de um procedimento, a palavra reservada PROCEDURE substitui PROGRAM, ampliando a "lista de parâmetros formais" de modo a incluir identificadores dos tipos de parâmetros. Assim, por exemplo,

```
PROCEDURE LINHAS (NUMLINHAS : INTEGER);
  VAR
    N : INTEGER;
  BEGIN
    FOR N := 1 TO NUMLINHAS DO
      WRITELN
END;
```

Ao END do procedimento segue-se ponto-e-vírgula, e não ponto-final, como no fim do programa. NUMLINHAS, o identificador de parâmetros formais, recebe o valor efetivo na instrução de chamadas:

```
LINHAS (5)
```

Esse procedimento trivial mostra-se muito útil quando queremos deixar várias linhas em branco (cinco, no caso) para facilitar a leitura, evitando seqüências contínuas de instruções WRITELN separadas por sinais de ponto-e-vírgula.

Âmbito

O exemplo precedente também serve como ilustração da confiabilidade do PASCAL. A variável FOR, para controle de loop, sempre é declarada como uma variável local estrita. Não se diz, portanto,

```
FOR NUMLINHAS := 1 TO NUMLINHAS DO
```

Embora isso seja aceitável num programa principal, a confiabilidade do loop não estará garantida enquanto o controlador do loop for não local, ou "relativamente global". Certamente você já passou horas depurando um programa em BASIC contendo uma instrução como

```
300 FOR N=1 TO T
400 GOSUB 2000
500 NEXT N
```

para descobrir, no final, que uma rotina — talvez empregada de modo indireto e condicional — usava a variável N com algum outro objetivo. O PASCAL não permite, em hipótese alguma, essa ambigüidade, possibilitando assim uma grande economia de tempo na elaboração dos programas. Qualquer identificador declarado no interior de um bloco opera numa "região" definida pelos limites do bloco. No entanto, esse "âmbito" (a parte do bloco a partir da qual o identificador é acessível) estende-se apenas de sua declaração até o final do bloco, e poderia ser limitado ainda mais por uma redefinição. No exemplo anterior, o N referido no procedimento LINHAS é um valor integer local, cancelando, a cada execução do procedimento, qualquer outra declaração relativamente global.

No esquema do programa principal (AMBITO), mostrado no quadro *acima*, a região das variáveis globais N e X corresponde a todo o programa. O âmbito de X vai desde sua declaração até o final do programa, excetuando-se o interior do procedimento A. O motivo está na existência de um outro X (SET OF CHAR) definido como identificador de tipo, cuja região é o bloco A. De maneira análoga, no interior de B, N refere-se ao parâmetro formal de B, e não à variável global.

Embora o âmbito tanto de A como de B corresponda a todo o programa, o de B só começa a partir de seu ponto de definição. Assim, enquanto for possível utilizar A no procedimento B, este será inacessível para A. Isso reforça a lógica rigorosa do PASCAL, impossibilitando a execução de qualquer programa até que tenha sido escrito. Por isso, as definições e declarações vêm, obrigatoriamente, na ordem CONST, TYPE, VAR, seguidas de procedimentos e funções, de acordo com os requisitos estruturais do programa. Muitos compiladores do PASCAL são de "passagem única" (lêem o código-fonte uma única vez), impossível sem uma rígida ordenação lógica.

O esforço adicional para declarar os dados locais em qualquer procedimento é amplamente recompensado. Uma das vantagens é a obtenção da modularidade por meio da transferência, para procedimentos, de todos os valores de dados sob a forma de parâmetros. Embora existam programas em PASCAL cujos procedimentos não apresentam nenhuma lista de parâmetros (todos os dados são acessados globalmente), essa não é uma boa prática de programação.

Observe que, assim como qualquer referência a X no interior de A implica o tipo X, o uso de

Âmbito abrangente

```
PROGRAM AMBITO;
VAR
    N : INTEGER;
    X : REAL;
PROCEDURE A (Y : REAL);
  TYPE
    X = SET OF CHAR;
  {ETC.}
PROCEDURE B (N : INTEGER);
  {ETC.}
BEGIN {AMBITO PRINCIPAL}
  ...
  A (SUCC (N)/3);
  B (N);
  A (X);
    {ETC.}
END.
```

No programa AMBITO, aqui delineado, as regiões e os âmbitos das diferentes variáveis e procedimentos são os seguintes:

Proc/Var	Região	Âmbito
A	Principal	A, B, principal
B	Principal	B, principal
N (global)	Principal	A, principal
X (real)	Principal	B, principal
X (no proc A)	A	A
N (no proc B)	B	B

N no interior de B refere-se a seu parâmetro local, deixando o valor integer global N temporariamente inacessível. A chamada B transfere o valor do global N como um parâmetro efetivo, designado por acaso com o mesmo nome em B, mas indicando uma variável diferente. Disso resultam vários pontos diferentes:

- Uma cópia local do valor transferido é feita na entrada de um bloco.
- Qualquer alteração desse "parâmetro de valor" no interior de seu bloco não altera o parâmetro efetivo transferido pela chamada.
- O valor transferido pode ser qualquer expressão do tipo adequado.

Qualquer instrução em um procedimento relaciona os parâmetros efetivos da chamada, os quais correspondem (quanto ao número, posição e tipo) à lista de parâmetros formais no cabeçalho. O mecanismo para transferência do valor de parâmetro pode ser considerado como uma inicialização do identificador de parâmetros formais para qualquer valor especificado na instrução de chamada. Assim,

```
LINHAS (SUCC (N + A) DIV 2
```

implica a seguinte instrução atributiva na entrada para LINHAS:

```
NUMLINHAS := SUCC (N + A) DIV 2
```

De uma base para outra

O programa CONVBASE exemplifica o uso de procedimentos com parâmetros. Você escolhe qualquer base entre 2 (binária) e 16 (hexadecimal), e os números decimais introduzidos serão apresentados na notação apropriada. Por exemplo, 32767 seria representado por 7FFF em hexadecimal, 77777 em octal ou 111111111111111 em binário. Se você quiser operar em sentido inverso, use o programa como modelo para, tomando um número em qualquer base (de 2 a 16), obter o resultado sob forma decimal. Que tal incorporar essa modificação como um procedimento do programa abaixo? Outro exercício seria imaginar um modo de fazer com que o programa opere com representações negativas. Para isso, devemos converter o número no seu complemento a 2. Ao nível de máquina, obtém-se isso pela negação do número e adição de uma unidade.

```
PROGRAM CONVBASE (INPUT, OUTPUT);
        {CONVERTE NUMEROS DECIMAIS PARA QUALQUER
          BASE DE BINARIO A HEXADECIMAL}
CONST
        COLUNAS    = 79; {PARA VIDEO/IMPRESSORA}

TYPE
        BYTE       = 0 .. 255;
        CARDINAL   = 0 .. MAXINT;

VAR
        NUMERO,
        BASE       : INTEGER;
        TROCAR,
        LEGAL      : BOOLEAN;

        {!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!}

PROCEDURE IMPREDIGITO (DIGITO : BYTE);
    {VALOR DE LISTA [CONTADOR] PASSADO A DIGITO}
  BEGIN
    IF DIGITO IN [0 .. 9]
      THEN
        WRITE (DIGITO : 1)
      ELSE              {REPRESENTAR COMO A .. F}
        CASE DIGITO OF
          10        : WRITE ('A');
          11        : WRITE ('B');
          12        : WRITE ('C');
          13        : WRITE ('D');
          14        : WRITE ('E');
          15        : WRITE ('F');
        END;            {CASE}
  END; {IMPREDIGITO}
        {!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!}

PROCEDURE    IMPRIME    (N          : CARDINAL;
                         BASE       : BYTE);
  CONST {TODOS ESTES DADOS SAO LOCAIS PARA IMPRIME}
        MAXDIGITOS = 32;
  TYPE
        LIMITES      = 1 .. MAXDIGITOS;
  VAR
        LISTA        : ARRAY [LIMITES] OF BYTE;
        INDICE       : LIMITES;
        CONTADOR     : BYTE;
  BEGIN
    CONTADOR := 0;

    REPEAT
        CONTADOR := SUCC (CONTADOR);
        LISTA [CONTADOR] := N MOD BASE;
        N := N DIV BASE

    UNTIL N = 0;
        {IMPRIME COMECANDO PELO MAIS SIGNIFICATIVO}
    FOR CONTADOR := CONTADOR DOWNTO 1 DO
        IMPREDIGITO (LISTA [CONTADOR]);
    END; {IMPRIME}
        {!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!}
  BEGIN {CONVBASE — PROGRAMA PRINCIPAL}

    REPEAT
      WRITELN ('ESCOLHA UMA BASE : ');
      WRITELN ('          2 .. 16          : COLUNAS);
      WRITELN ('(QUALQUER OUTRA ENCERRA)' : COLUNAS);
      WRITE ('BASE?');
      READ (BASE);
      LEGAL := BASE IN [2 .. 16];

      IF LEGAL THEN
        BEGIN
          WRITE ('NUMERO (0 TROCAR BASE)?');
          READ (NUMERO);
          TROCAR := NUMERO <= 0;

          WHILE NOT TROCAR DO
            BEGIN
              WRITE (NUMERO : COLUNAS DIV 2,
                'TO BASE', BASE : 1, 'E');
              IMPRIME (NUMERO, BASE);
              WRITELN;
              WRITE ('NUMERO?');
              READ (NUMERO);
              TROCAR := NUMERO <= 0;
            END
        END

    UNTIL NOT LEGAL

  END.
```

SUMÁRIO

VOLUME 1

APLICAÇÕES

A máquina de hoje. 5
A escrita na tela. 17
Informação em cadeia. 37
Descobrindo a senha. 57
Computador espião. 77
Os pesos leves. 97
Fatos nas pontas dos dedos. 117
Superando limitações. 137
Processador de idéias. 157
Paleta eletrônica. 177
Dados a distância. 197
Xadrez nos chips. 217
Homens e máquinas. 237
O cérebro artificial. 257
Controle por computador. 277
Túnel do tempo. 297

CHIPS & BYTES

Segunda mão. 310

CIÊNCIA DA COMPUTAÇÃO

A álgebra das decisões. 28
A estrutura dos blocos de adição. 48
Refinando o processo. 92
A lógica da programação. 108
Siga o mapa. 154
Mapas e circuitos. 168
A resposta lógica. 208
Mudanças de código. 232
Métodos alternativos. 272
No nível. 292

LINGUAGENS

Códigos lógicos. 7
Os passos da tartaruga. 30
A arte da tartaruga. 50
Duplo desempenho. 70
Divida e governe. 79
Perdida no espaço. 82
Calculando com o LOGO. 110
Poesia aleatória. 132
Quem é o culpado? 148
Repita o desempenho. 170
Primeiras palavras. 180
Dados e decisões. 211
1 + 2 = 20? 220
Em conjunto. 251
Registre. 260
Dimensões matriciais. 288
Lingüística estrutural. 312

OFICINA

Ferramentas essenciais. 68
A troca do chip. 128
Teste de bancada. 192
Circuitos e componentes. 244

PERFIL

Revolução nos circuitos. 36
Os dissidentes. 56
Terminal bancário. 76
Investindo no futuro. 96
O início do jogo. 115
Programas compatíveis. 135
Música e números. 156
Ousados conservadores. 175
Soluções integradas. 196
Elegância italiana. 216
Objetivo brasileiro. 236
Radiocontato. 256
Eletrônica popular. 276
Lógica empresarial. 296

PERIFÉRICOS

Visão periférica. 14
Escolha de arquivos. 20
Dados na tela. 40
Cópias quentes. 59
Impressoras matriciais. 88
Na mira do rifle. 100
Tudo sob controle. 120
Trilhas espirais. 151
Primeiras impressões. 160
Mouse e iconografia. 184
O poder da flor. 200
Desempenho versátil. 240
Traços e dígitos. 263
Printer/plotters. 282
O melhor de dois mundos. 301

PROJETOS DE PROGRAMAS

A proxima letra. 24
Somas mágicas. 44
Gráficos. 64
Ordene os dígitos. 85
Magia Animal. 104
Pedalando. 124
Torre de Hanói. 142
O pouso do módulo. 164
A terceira dimensão. 190
Psiquiatra eletrônico. 204
Cores e números. 230
Um novo ângulo. 242
Códigos de comando. 268
A revolução cubista. 270
Travessia do Deserto. 280
Teste de reflexos. 300

RAIO X

As famílias de micros. 9
IBM PC. 26
Apple II *c*. 46
Micros Sharp. 66
Macintosh. 86
Apricot. 106
Plus/4. 126
Alfa 3003. 146
I-7000 PCxt. 166
TK 90X. 186
Grafix 100MX. 206
Wafadrive. 225
Câmara escura. 246
Compaq Plus. 266
BBC modelo B. 285
Grafpad. 305

SOFTWARE

Lições e desafios. 22
Memória organizada. 42
Arquivo organizado. 62
Arquivos seqüenciais. 90
Seleção aleatória. 102
Posições-chaves. 122
Arquivos na RAM. 140
A soma das partes. . . . . 16[illegible]
Programas integrados. . . . . 18[illegible]
1-2-3: macrocomandos. 202
Controle total. 228
A produção de softwares. 248
Aplicativos. 291
Organização de dados. 303

TÉCNICAS DE PROGRAMAÇÃO

Métodos de trabalho. 8
Plano de ação. 34
Cuidados com o estilo. 54
Conceitos básicos. 75
Linhas de loop. 95
Decisões cruciais. 113
Peças de quebra-cabeça. 130
Supersecreto. 144
Fontes de erros. 173
O fator humano. 194
Ajuda em linha. 234
Faça sua escolha. 254
Biblioteca circulante. 274
Métodos de otimização. 294
Check-up de rotina. 308

```
230 PRINT SPC(10)"/------*------*------"CHR$
    (92)
240 PRINT SPC(10)"I      I      I      I"
250 PRINT SPC(10)"I      I      I      I"
260 PRINT SPC(10)"I      I      I      I"
270 PRINT SPC(10)"*------*------*------*"
280 PRINT SPC(10)"I      I      I      I"
290 PRINT SPC(10)"I      I      I      I"
300 PRINT SPC(10)"I      I      I      I"
310 PRINT SPC(10)"*------*------*------*"
320 PRINT SPC(10)"I      I      I      I"
330 PRINT SPC(10)"I      I      I      I"
340 PRINT SPC(10)"I      I      I      I"
350 PRINT SPC(10)CHR$(92)"------I------I--
    ----/"
360 FOR R=0 TO 2
370 FOR C=0 TO 2
380 VTAB 5+4*R:HTAB 12+6*C
390 PRINT MID$(IL$,3*R+C+1,1)
400 NEXT C,R
410 GOSUB 640
420 RETURN
430 REM ** ENTRADA DE INSTRUCOES **
440 X=PEEK(-16384):REM  TECLA
450 IF X<128 THEN 440
460 POKE -16368,0:REM   RESET
470 IN=0
480 FOR I=1 TO 9
490 IF CHR$(X-128)=MID$(IL$,I,1) THEN IN
    =I
500 NEXT I
510 IF IN=0 THEN 440
520 RETURN
530 REM ** MOVIMENTO **
540 R=INT((IN-1)/3)
550 C=IN-3*R-1
560 FOR I=0 TO 2
570 IF B(R,I)=6 THEN B(R,I)=0
580 B(I,C)=B(I,C)+1
590 IF B(I,C)=6 THEN B(I,C)=0
600 NEXT I
610 B(R,C)=B(R,C)-1
620 IF B(R,C)=-1 THEN B(R,C)=5
630 RETURN
640 REM ** SAIDA **
650 NZ=0
660 FOR R=0 TO 2
670 FOR C=0 TO 2
680 VTAB 6+4*R:HTAB 14+6*C
690 PRINT B(R,C);
700 NZ=NZ+(B(R,C)=0)
710 NEXT C,R
720 RETURN
730 REM ** PROGRAMA PRINCIPAL **
740 FOR TT=1 TO 9
750 IN=INT(RND(1)*9+1)
760 GOSUB 530
770 NEXT
780 GOSUB 190
790 GOSUB 430
800 GOSUB 530
810 NM=NM+1
820 GOSUB 640
830 IF NZ<9 THEN 790
840 VTAB 20:HTAB 8
850 PRINT "CONSEGUIU EM "NM" MOVIMENTOS
    !"
860 END
870 REM ** INSTRUCOES **
880 HOME :VTAB 2:HTAB 15:PRINT "TUDO ZER
    O"
890 VTAB 10:PRINT "DESEJA INSTRUCOES (S/
    N) ";
900 GET A$:IF A$<>"S" THEN RETURN
910 PRINT
920 VTAB 6:PRINT "O OBJETIVO DESTE  JOGO
     E' CONSEGUIR FA-"
930 PRINT "ZER COM QUE TODAS  AS COLUNAS
     DE UM QUA-"
940 PRINT "DADO  CONTENHAM ZEROS.
    "
950 PRINT :PRINT "CADA COLUNA CONTEM TRE
    S LETRAS REPRESEN-"
960 PRINT "TANDO  CADA LOCAL  NESTA.  VO
    CE TERA QUE"
970 PRINT "DIGITAR  A LETRA CORRESPONDEN
    TE A CELULA "
980 PRINT "DA COLUNA QUE SE DESEJA MOVIM
    ENTAR MENOS"
990 VTAB 23:PRINT "QUALQUER TECLA PARA C
    ONTINUAR ";
1000 GET A$:PRINT :RETURN
```

ORDEM

PARA LINHA SINCLAIR

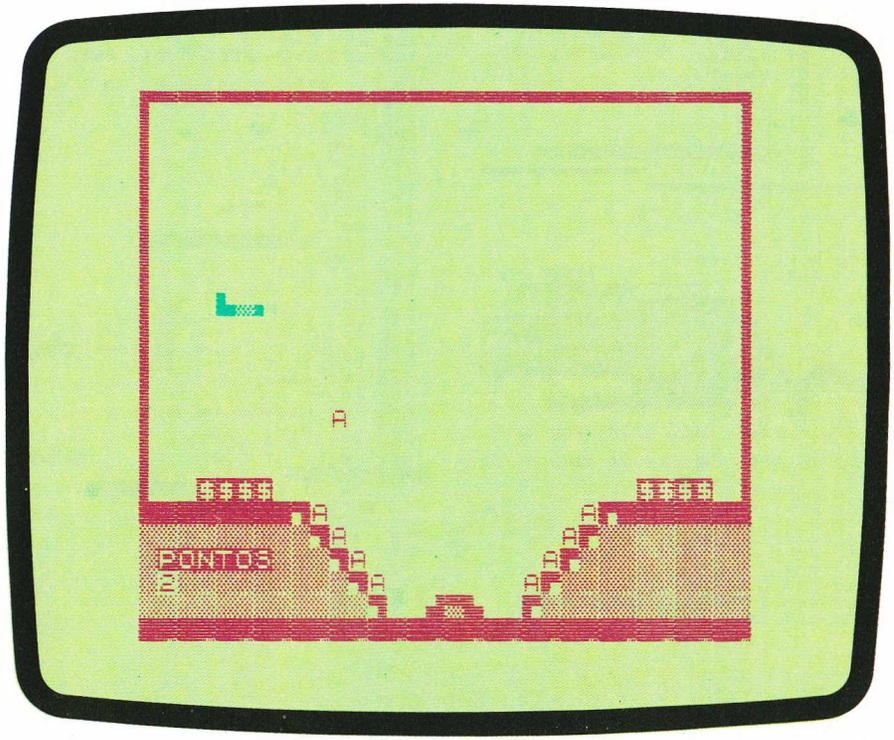
A tela mostra uma seqüência aleatória de nove algarismos. Seu objetivo é arranjá-los de maneira crescente e, para tanto, você dispõe de um único comando, com o qual inverte a ordem de quantos números quiser a partir do primeiro. Imagine, por exemplo, que na tela esteja a seqüência 654813279. O programa lhe pergunta "Quantos?" e você digita [3]. Então, os três primeiros algarismos se invertem, e a seqüência fica 456813279. Daí em diante, procure inverter a ordem da melhor forma a fim de chegar à seqüência correta.

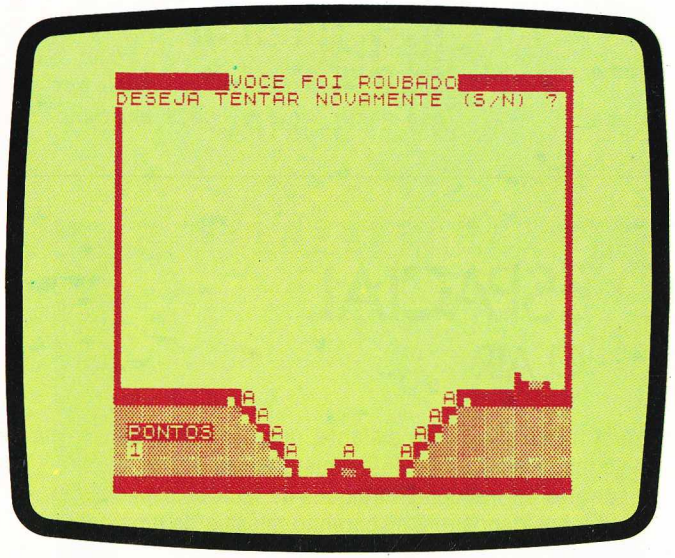
```
10 REM
20 REM        ORDEM
30 REM  PARA A LINHA SINCLAIR
40 REM
50 FAST
60 LET T=0
70 DIM A$(9)
80 LET A$="000000000"
 90 FOR I=1 TO 8
100 LET R=INT (9*RND)+1
110 IF A$(R)>"0" THEN GOTO 100
120 LET A$(R)=CHR$ (28+I)
130 NEXT I
140 FOR I=1 TO 9
150 IF A$(I)="0" THEN LET A$(I)
="9"
160 NEXT I
170 SLOW
180 PRINT AT 1,0;A$
190 LET T=T+1
200 PRINT AT 3,0;"QUANTOS ? ";
210 INPUT X
220 PRINT X
230 IF X>9 THEN GOTO 200
240 FOR I=1 TO INT (X/2)
250 LET Y$=A$(I)
260 LET A$(I)=A$(X-I+1)
270 LET A$(X-I+1)=Y$
280 NEXT I
290 PRINT AT 1,0;A$
300 IF A$<>"123456789" THEN GOT
O 190
310 PRINT AT 5,0;"VOCE ORDENOU
EM ";T;" TENTATIVAS"
```

LADRÃO ESPACIAL

PARA LINHA SINCLAIR

Esta aventura se passa num futuro não muito distante, quando a falta de energia em nossa galáxia tornou-se insuportável. Devido a seus grandes depósitos de urânio, a Terra passou a ser alvo de freqüentes ataques de ladrões espaciais. Sua missão é proteger um desses depósitos, disparando mísseis de uma base, a fim de impedir que as naves piratas desçam e carreguem o urânio.
Seus comandos são: [0], dispara os mísseis; [5], esquerda; e [8], direita.
Na tela, você acompanha a contagem de pontos.





Defesa impossível

Ladrão espacial é um jogo em que o participante sempre acaba perdendo, como nos fliperamas, pois o número inesgotável de naves inimigas torna impossível a defesa permanente dos depósitos.

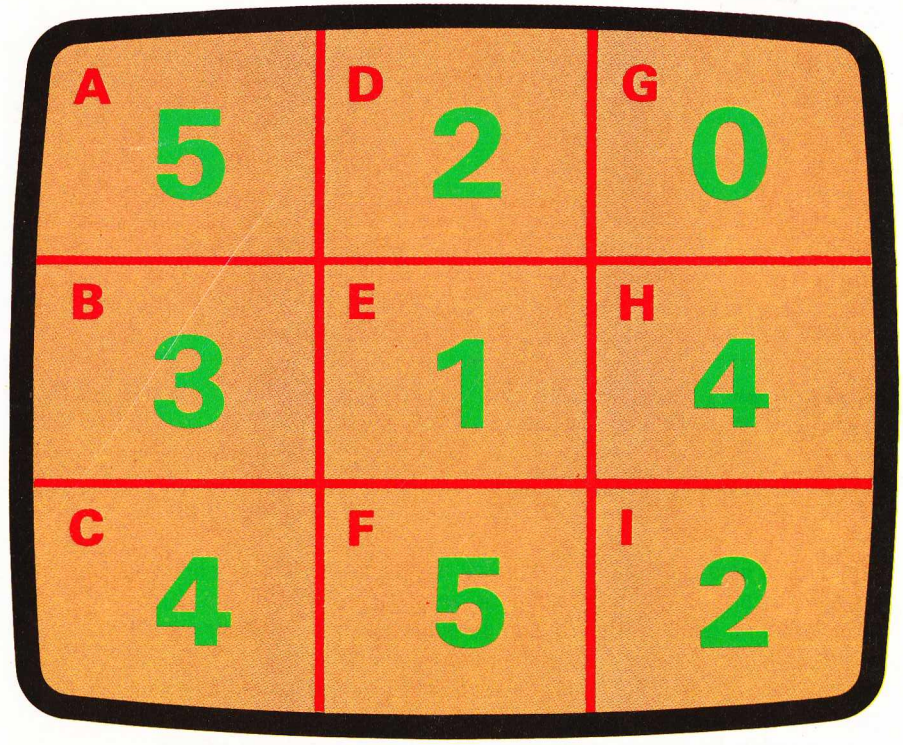
O interessante do jogo está em superar as marcas anteriores, atingindo maior número de naves a cada vez.

```
  10 REM
  20 REM   LADRAO ESPACIAL
  30 REM  PARA A LINHA SINCLAIR
  40 REM
  50 CLS
  60 SLOW
  70 LET G=PEEK 16396+256*PEEK 1
6397
  80 LET A=G+34
  90 LET B=A+1
 100 LET C=G+710
 110 LET D=C
 120 LET E=2
 130 LET F=0
 140 REM  MONTAGEM TELA
 150 PRINT AT 0,0;"
                  ";
 160 FOR X=1 TO 17
 170 PRINT AT X,0;"
                  ";
 180 NEXT X
 190 POKE 16418,0
 200 PRINT AT 17,0;"
                  ";
 210 REM  ACAO CENTRAL
 220 LET A=A+2
 230 IF PEEK (A+1)=133 THEN LET
A=A+38
 240 IF PEEK (A+1)=136 THEN GOTO
 660
 250 POKE B,0
 260 POKE (B+1),0
 270 POKE (B+2),0
 280 POKE A,130
 290 POKE (A+1),9
 300 POKE (A+2),4
 310 LET B=A
 320 REM  LEITURA TECLAS
 330 LET L$=INKEY$
 340 LET C=C+(L$="8")-(L$="5")
 350 IF L$="0" AND E=2 THEN LET
E=0
 360 IF E=2 THEN GOTO 210
 370 IF E=0 THEN LET C=G+710
 380 IF E=0 THEN LET D=C
 390 LET E=1
 400 LET C=C-33
 410 LET C=C+(PEEK C=5)-(PEEK C=
133)
 420 IF PEEK C=131 THEN GOTO 480
 430 IF PEEK C=9 THEN GOTO 520
 440 POKE D,0
 450 POKE C,38
 460 LET D=C
 470 GOTO 210
 480 POKE D,0
 490 LET E=2
 500 POKE (G+710),38
 510 GOTO 210
 520 REM  EXPLOSAO
 530 POKE D,0
 540 LET E=2
 550 LET X=(A-G-1)
 560 LET LINHA=INT (X/33)
 570 LET COLUNA=X-(LINHA*33)
 580 PRINT AT LINHA,COLUNA;"
 590 PRINT AT LINHA,COLUNA;"XXX"
 600 PRINT AT LINHA,COLUNA;"
 610 LET F=F+1
 620 PRINT AT 21,1;F
 630 LET A=G+34
 640 LET B=A+1
 650 GOTO 500
 660 PRINT AT 0,0;"
                  VOCE
FOI ROUBADO
                  "
 670 PRINT AT 0,0;"
                  VOCE
FOI ROUBADO
                  "
 680 PRINT "DESEJA TENTAR NOVAME
NTE (S/N) ?"
 690 LET L$=INKEY$
 700 IF INKEY$="S" THEN RUN
 710 IF INKEY$<>"N" THEN GOTO 66
0
```


TUDO ZERO

PARA LINHA APPLE

Neste quebra-cabeça, nove quadrados estão dispostos em três linhas e três colunas. Em cada um há um número de 0 a 5 e uma letra. Quando você tecla essa letra, os outros dois quadrados da mesma coluna têm seus números aumentados uma unidade. Vamos supor, por exemplo, que o quadrado contenha o número 5, o B 3 e o C 4: digitando [B], A passará a 0, C a 5 e B não se alterará. Seu objetivo neste jogo consiste então em fazer com que todos os quadrados contenham 0, pressionando as teclas corretas.




```
100 REM ********************************
110 REM *      PARA LINHA APPLE        *
120 REM *                              *
130 REM *          TUDO ZERO           *
140 REM ********************************
150 IL$="QWEASDZXC"
160 DIM B(2,2)
170 GOSUB 870
180 GOTO 730
190 REM ** TELA **
200 HOME
210 PRINT SPC(15)"TUDO ZERO"
220 VTAB (4)
```